ENHANCING TEXT-BASED SEARCH FOR WEB IMAGES

ENHANCING TEXT-BASED SEARCH FOR WEB IMAGES

by

Rami Abdullwahab Al-Hailat

Dr. Natheer Khasawneh

Dr. Ahmad Al-Hammouri

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of M.Sc.

in Computer Engineering

at

Faculty of Graduate Studies

Jordan University of Science and Technology

January, 2010

ENHANCING TEXT-BASED SEARCH FOR WEB IMAGES

by

Rami Abdullwahab Al-Hailat

Signature of Author	....
Committee Members	**Signature and Date**
Dr. Natheer Khasawneh (Chairman)	....
Dr. Ahmad Al-Hammouri (Co-Advisor)	....
Dr. Eyad Taqieddin (Member)	....
Dr. Sulieman Bani-Ahmad (External Examiner)	....

January, 2010

DEDICATION

To My Great Father and Mother for Their Passion

To My Brother Mohammad and my Sisters, Roba, Rana, Rawan, Razan, Reham, Jasmin and Ola

ACKNOWLEDGMENTS

I would like to express my deepest thanks to my advisors, Dr. Natheer Khasawneh and Dr. Ahmad Al-Hammouri, for their professional, invaluable support at all the time during my graduate education at Jordan University of Science and Technology (JUST). I would also like to thank Dr. Eyad Taqieddin and Dr. Sulieman Bani-Ahmad for serving on my master defense committee. I would also like to thank the friends who cares about me, and keep asking about my progress in this research, thank you Bawaneh, Hazem and Nayef. Finally, I would like to thank my parents, for their continuous support and encouragement, you inspired me. The support of the Jordan Armed Force, from my undergraduate education till now is greatly appreciated.

That person who shall pursue the path of knowledge, God will direct him to the path of Paradise; and verily the superiority of a learned man over an ignorant worshiper is like that of the full moon over all the stars.

Prophet Muhammad, peace be upon him. Saheh Abi Dawood 2

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

ENHANCING TEXT-BASED SEARCH FOR WEB IMAGES

by

Rami Abdullwahab Al-Hailat

Recent studies state that there will be more than 300 million digital-image capture devices in use worldwide [7], capturing about 29 billion digital pictures, most of which will be organized in some kind of multimedia repositories and published to the World Wide Web. Many of these images need to be stored, sorted and retrieved later on. From here a systematic approach is crucial to help ordinary users find useful information on the Web, especially of we know that 15-16% of web searches are for images [8]. Different approaches were invented for searching and fetching web images, most of them use the textual context of web pages to judge image relevance or irrelevance to user query. Other methods uses other means in searching images over the web. For example, they use the internal and external image contents and features, such as, image histogram, texture, patterns and boundary descriptors. These different approaches have several advantages and disadvantages . This Thesis proposes to enhance image search by applying a new technique that uses the physical distance between an image on the web page and its surrounding textual terms. The proposed approach shows a considerable enhancement on the process of ranking image search result.

Chapter 1

Introduction

In this thesis, we introduce a novel algorithm for ranking and ordering image search result from a popular search engine. In this chapter, we start with the motivation of this work. Then, we describe currently used image retrieval methods. Finally, we state the goals of the thesis and we summarize our contributions.

1.1 Motivation

The Dramatic revolution of high-density and low-cost storage devices and local drives supported by the invention of high resolution digital cameras facilitated people to the ability to capture, share and store diverse of digital images [9, 10]. The widespread of Internet coupled with the easiness of Web hosting, make the World Wide Web (WWW) an important information resource for huge number of images scattered in different geographical locations [11].

Images are mainly embedded inside Html documents, which dominate the Web as a formatting language. Images inside Html documents are tagged and are usually labeled with some meaningful text that reflects the content of these images. A systematic approach is needed to help ordinary users find useful information, or images on the Web. So, different types of image search engines already exist, such as Google, Yahoo and Microsoft's Bing. An image search engine typically consists of two steps, crawling and ranking. A crawler, or an indexing engine, recursively visits Web pages, indexes their contents, and extracts contained links to further crawl other pages.

An index database is then built and organized as an inverted file [12] to facilitate future search and recall of information, (for more information about it see section 2.1.2).

The second step in any image search engine is the ranking or scoring. This step uses the already constructed index file to retrieve results based on a specific query. Results are usually ranked based on the degree of potential matching of user submitted query and data in the index file.

The indexing stage processes the crawled Html pages to extract the set of features that will make the construction of index database more efficient and accurate in querying, retrieving and ranking possible results. Different crawlers extract different features list based on the crawling type.

1.2 Image Retrieval Types

Currently, there are two main Image Retrieval methods: Text-based Image Retrieval (TBIR) and Content-based Image Retrieval (CBIR). These two techniques have significant limitations that degrade image retrieval performance when compared to searching for text documents.

TBIR usually performs search given one or more keywords (terms). An old version of this type uses boolean-like query on keywords, which simply finds an identical occurrence of the given term. Modern version use more complicated procedures taking into account natural language processing like stop-word removal and stemming [13].

CBIR differs from the TBIR in two ways. First, CBIR expects an image or a sketch as an input to the system, so that the system can query for some similar images from the Web. Second, CBIR extracts the visual features of the indexed Web-pages images, and builds a visual feature space so that each image is represented by a feature vector on that space.

CBIR implementations decide similarity as a measure of the distance between two

candidate vectors (images features) [14, 15]. CBIR usually relies on colors because of the simplicity in extracting intensities from image pixels. Texture is also used but less widely than colors; then comes shapes, spatial layout, and faces.

There is also a mixed technique, called Hybrid-based Image Retrieval (HBIR) which combines both (CBIR) and (TBIR) processing with different variations. An example of a system that supports such processing is the COWEB [5].

Recently, there have been standards to formalize Html pages design in order not to omit some important fields of Web pages that search engines rely on. This will definitely support text-based searching and indexing in general as it improves semantic and logical search. An important issue in image search is the scalability of the used method to accommodate parsing and indexing of millions of Web pages and to recursively processing page's meta-data, annotations and extracting colors, shapes and textures features. A detailed description of these techniques is presented in Chapter 2.

1.3 Objectives

The main aim in this thesis is to enhance the ordering and ranking of image search method by taking into consideration the physical distance between an image and a keyword in the Web page. Physical distance is the actual distance in pixels between two locations inside a web-page, the measurement is carried on after opening and rendering a web-page, this will produce a real physical distance, which in turn expected to reduce the number of irrelevant images in the returned search result. This is valid as text within a search query is associated more strongly to an image inside the same document if the term is closer to an image than any other images in that page [16, 17]. To verify this, we compare the results of our system to those of a well known image search engine , such as Google.

1.4 Summary and Contributions

Existing search algorithms try to simulate visual properties of Web pages and use a textual Html structure properties. For example, in [16, 11], the authors represented the visual content of pages using pure text. This operation is error prone because Html is not a strongly typed programing language, and a small error will result in different layouts of the same page [18]. Also, differences in rendering exist in different Web browsers.

While the text might include useful information related to an image, selecting which part better describes an image is not a trivial task [19]. Our proposed work tags an image with the a keyword that is probably characterizes it.

First, our system takes a query term from the user and consults an image search engine to obtain several candidate web pages (WP_{URL}). Our system renders each web page (WP_{URL}) creating a complete Web page Image (WP_C), and finds the coordinate of each embedded image (RP_{URL}) inside WP_C.

Then, the system relates this RP_{URL} to the surrounding text in order to retrieve a special weight or score. This whole process is carried on for each result comes from search engine. Then a score ordering algorithm is executed to obtain results (see Figure 1.1), Finally, it worth to say that our algorithm is supposed to work after integrated to some search engine, so that all the steps applied will be off-line.
The following points summarize the contributions of this work:

1. Presenting a novel algorithm to enhance ranking of image-search results.

2. Applying pattern matching and client scripting in measuring the physical distance between keywords and images in a web page.

3. Integrating the presented algorithm with Google image search engine.

4. Building an interface to evaluate the results of the presented algorithm

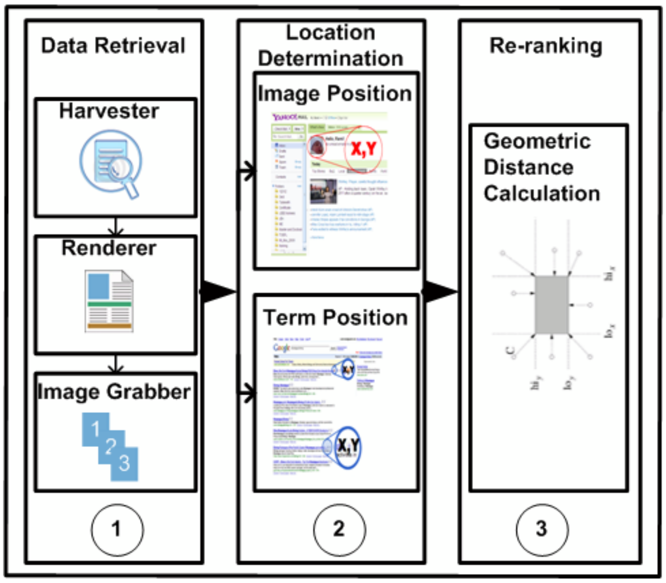


Figure 1.1: *Proposed algorithm high-level Architecture*

1.5 Outline of thesis

The list below presents the major outline of the chapters that build up this thesis:

- **Chapter 2, Previous Work** provide a literature review in the field of image retrieval, supported with examples of related work and well known retrieval types, e.g., Text-based and Content-based.

- **Chapter 3, Employing Physical Distance In Re-ranking Image-Search Result** describes the proposed physical distance algorithm, and explains its modules and the cooperation between them, then, we end the chapter with an illustration example, demonstrating the whole idea.

- **Chapter 4, Results and Discussion** provide the experiments setup and methodology followed in this research, followed by data sets used in validating distance algorithm, and finally, a description of the experiment result.

- **Chapter 5, Conclusion and Future Work** concludes our work, and considers possible future research, generated by this thesis.

Chapter 2

Previous Work

Early approaches for image search and retrieval were text-based approaches [19]. Text based approaches, such as [11, 19, 20, 21], use description and some meta-data that were manually appended to images. Searching for images is a matter of issuing direct database queries. However, manually describing and annotating images became infeasible as mistakes in description usually happen. Thus, interest in image retrieval shifted toward content-based approaches, which depends on extracting image features, such as color, texture, or shape [5, 22].

The main problem of CBIR systems is the semantic gap between the image content represented with low-level visual features and the textual description given by humans [23]. Therefore, text-based searching still represents more semantic and logic in contrast to the content-based search. CBIR employs features of images that can be used to classify images, choosing these features are not a trivial task, as it need to decide which feature will the system adopt to classify an image content based on, especially if the classified images are with different content.

In this chapter we review and classify TBIR and CBIR previous work.

2.1 Text-based Image Retrieval (TBIR)

A TBIR system is usually built from a set of interacting modules: one for gathering data, others for processing these data and extracting useful properties facilitating simple retrieval in future, we will discuss these modules next in more detail.

2.1.1 Crawler

Any TBIR system traverses the Web, from some starting Uniform Resource Locater (URL), and navigates between published web documents; see Figure 2.1. Then, the TBIR system constructs an index of the visited pages, especially those containing images for future query and search operation.

The crawler simply sends HTTP requests for documents hosted on web servers scattered over the Internet, just as a Web browser does when the user clicks on a link. In other words, the crawler automates the process of following links. After that, it parses for any included hypertext links pointing to other web pages. A link depth can be specified in order to navigate to a limited number of Web pages. Then the crawler follow these new pages for new links, and the operation is repeated recursively.

Extracting Images

After the Web page content is downloaded by the crawler, parsing of the Html tags starts immediately and special tags are inspected, such as IMG and IMAGE tags. The parser navigates the downloaded page and anytime it finds such tags, it collects all the required attributes that this tag may carry, e.g, *alt*[1], width, height, style and image captions or names. Some parsers also take into account near heading tags (H1, H2). Images inside web pages often lack the alt tag, a meaningful name, and context, but the text surrounding images is always available.

Extracting Words

In order to analyze content, the Html parser parses the document and extracts both images and words. Words are collected in a data structure for next processing and analyzing phase.

Stemming, Stop words and Synonymy Processing

During analyzing of words, text-based search engines often use a stemming functionality. Stemming is suffix stripping based on some rules, for example catty becomes cat

[1] The *alt* attribute usually contains a short description of the image. This attribute is used when the user, for some reason, cannot see the image. See `http://www.w3schools.com/tags/att_img_alt.asp`.

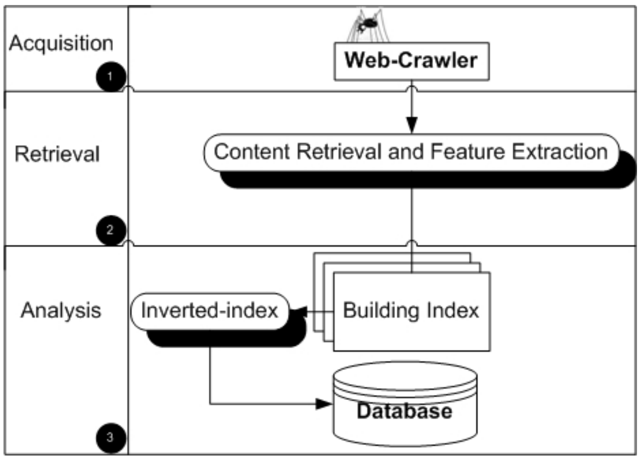


Figure 2.1: *Common TBIR components*

after stemming and playing becomes play. On the other hand, very frequent words contain little information and can be safely removed. These words are specific to each language, for example, in English, the words a, an, and, are, as, at, be are frequent and considered stop words, so they have to be removed from the analyzed word list because there is no value in processing them.

The synonym[2] database provided by the Word-net [3] is widely used for English language to build an index of synonyms. So, when a query is issued for a word such as car, a search engine also processes and modifies the query to also accommodate car synonyms, e.g., automobile.

After the crawling step, we need to store collected information into some data structure, a data persistence structure is explained next; see Figure 2.2.

2.1.2 Data Persistence Structure

After the input has been parsed and analyzed, it is ready to be added to the inverted index. The inverted index is a data structure that makes efficient use of disk space while allowing quick keyword lookups and retrieval. The constituent of this index is processed words (tokens) extracted from the input Web pages (documents). This

[2]two words that can be interchanged in a context.
[3]http://wordnet.princeton.edu/

structure uses tokens as lookup keys instead of treating documents as the central entities. In other words, instead of trying to answer the question "what words are contained in this document?", the structure is optimized for providing quick answers to "which documents contain word x?" [12].

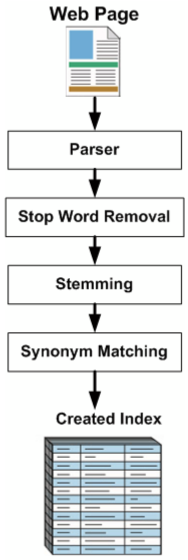


Figure 2.2: *Retrieval inner steps*

Inverted Index

As stated previously, the structure of the inverted index saves space and improves lookup time. Figure 2.3 shows how each document content is placed inside this index: each word is saved inside the index after applying stemming and stop word removal which shrink tokens number. The token saved represents the key of the table, and the value of this key represents the document that the token occurs inside it. Also, the table contains the word position and the frequency of occurrences. In Figure 2.3, if the user search for a word "sky" the inverted index returns that this word occurs inside two documents: it occurs inside document 2 at the position 8 and inside document 3 at the position 3. A more complex search query with more than one term will produce a ranked document result based on the closeness of the search term and the text inside document.



ID	Term	Document: Position
1	best	2:3
2	blue	1:3, 3:2
3	bright	1:2, 3:5
4	butterfly	1:4
5	breeze	1:8
6	forget	2:5
7	great	2:7
8	hangs	1:5
9	needs	3:8
10	retire	2:11
11	search	3:10
12	sky	2:8, 3:3
13	wind	2:14

Figure 2.3: *Example of constructed inverted index, adapted from Mac OS reference library[1]*

2.1.3 Weighting Mechanism

This step determines each image location in the page, and recognizes surrounding text, This is important in weighting how each term is relevant to a specific image. In order to distinguish between a relevant result and non-relevant images when searching the index, it is mandatory to adopt a scoring technique that eliminates or boosts the rank of a document based on a set of properties an image has; see equation 2.1.

$$score = \sum_{t} tf(t\ in\ d) \cdot idf(t) \cdot boost(t.field\ in\ d) \cdot lengthNorm(t.field\ in\ d) \tag{2.1}$$

Where $tf(t\ in\ d)$ is the term frequency factor for the term (t) in the document (d), *idf* is the inverse document frequency of the term, *boost* is field boost as set during indexing, this field can be used to bias some field to have more inference to score if it happen to be inside the searched document, and $lengthNorm$ is normalization value of a field, this depends on the number of terms a field contains.

The following section presents examples of previous work mainly based on Text-based Image Search.

2.1.4 Block-based Web Search

Many authors realize the relationship between the searched terms and its location inside the Web page, and they usually refer to it as a passage retrieval. Passage retrieval is a research topic with long history in the Information Retrieval that addresses the shortcomings and erroneous decisions made based on whole-document ranking [17, 24].

Block-based Web Search and Vision-based Page Segmentation Algorithm (VIPS) explore the use of a new page segmentation algorithms to partition Web pages into blocks and investigate how to take advantage of block-level evidence to improve retrieval performance in the Web context [16, 2]. Authors investigated four types of segmentation methods, including fixed-length page segmentation, DOM-based page segmentation, vision-based page segmentation, and a combined method which integrates both semantic and fixed-length properties. The basic idea is to segment page into different blocks because each page usually contains different topics and logical entities not a single one. So, treating a page as a single semantic unit might be misleading. If distribution of some terms in a specific region or segment occurs in high density, then this region is highly related to these terms, than considering the whole page as one segment.

Fixed-length Page Segmentation (FixedPS) [16] segments pages based on a predefined windows length of (200-250) words. The system starts counting words from the first occurrence of a specific term. For Web documents, fixed-length page segmentation is the same except that all the Html tags and attributes are ignored. Despite its simplicity, authors find that fixed-length segmentation is very robust and effective for improving performance, particularly for collections with long or mixed-length documents.

DOM-based Page Segmentation (DomPS) relies on the structure of a Web page as well as its presentation (see Figure 2.4). DomPS converts the page to a tree where each Html tag is represented by a node inside it. DomPS segmentation is based on the syntactic structure comes from Html standards.

A few approaches use this idea to segment web pages, while others use some Html tags, such as $<TITLE>$, $<P>$, $<H1>$ .... $<H3>$ and $<META>$ as candidate blocks. However the results are not encouraging (see section 2.1.5). DOM is a linear structure in that visually adjacent blocks at rendering time may be far from each other in the textual structure and might be departed wrongly.

```
<Table>
  <Tr>
     <Td>a</Td>
     <Td>b</Td>
  </Tr>
  <Tr>
     <Td>c</Td>
     <Td>d</Td>
  </Tr>
</Table>
```

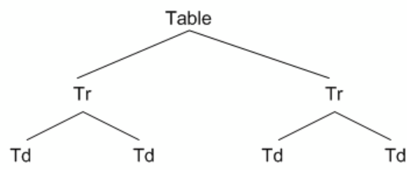


Figure 2.4: *An example of DOM structure for Html portion*

Vision-based Page Segmentation (VIPS) . Visual treatment of Web pages simplifies distinguishing different parts and blocks of the page. For the sake of easy browsing and understanding, a closely packed block within the Web page is much likely about a single semantic. Similar to semantic passages, the blocks obtained by VIPS are based on the semantic structure of Web pages. VIPS produces blocks based on the visual cues of Web pages. This method simulates how a user understands Web layout structure based on his or her visual perception [2]. The DOM structure and visual information(Html tag libraries and standards) are used iteratively for visual block extraction to produce a vision-based content structure.

In Figure 2.5, the vision-based content structure of a sample page is illustrated. Visual blocks are detected as shown in figure 2.6. It is an approximate reflection of the semantic structure of the page. VIPS differs from DOM based segmentation in that VIPS aggregates nodes with different granularities, while DOM structure contains nodes in a tree shaped representing a direct relation between different nodes in a child parent manner.

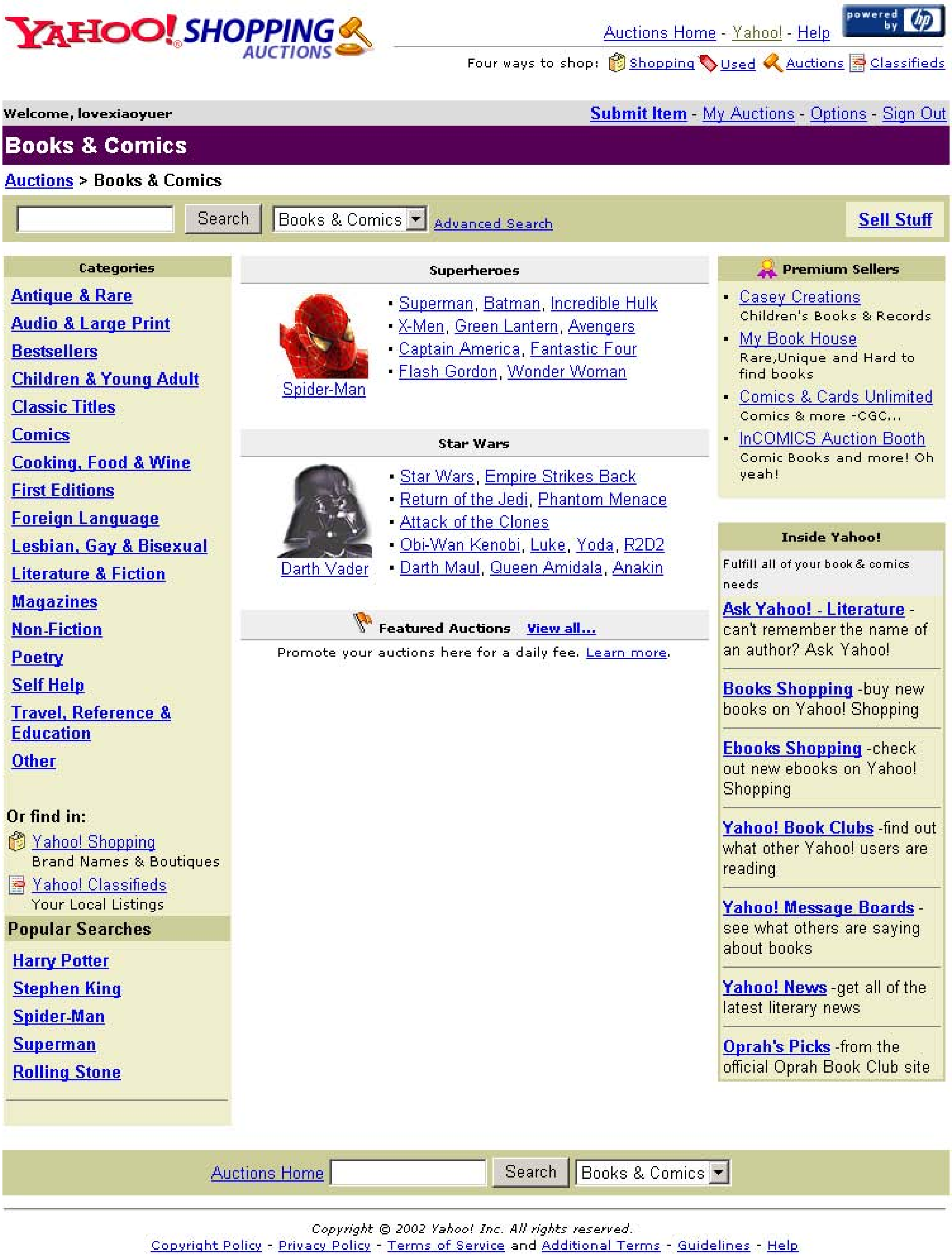


Figure 2.5: *An example of a web page for VIPS*

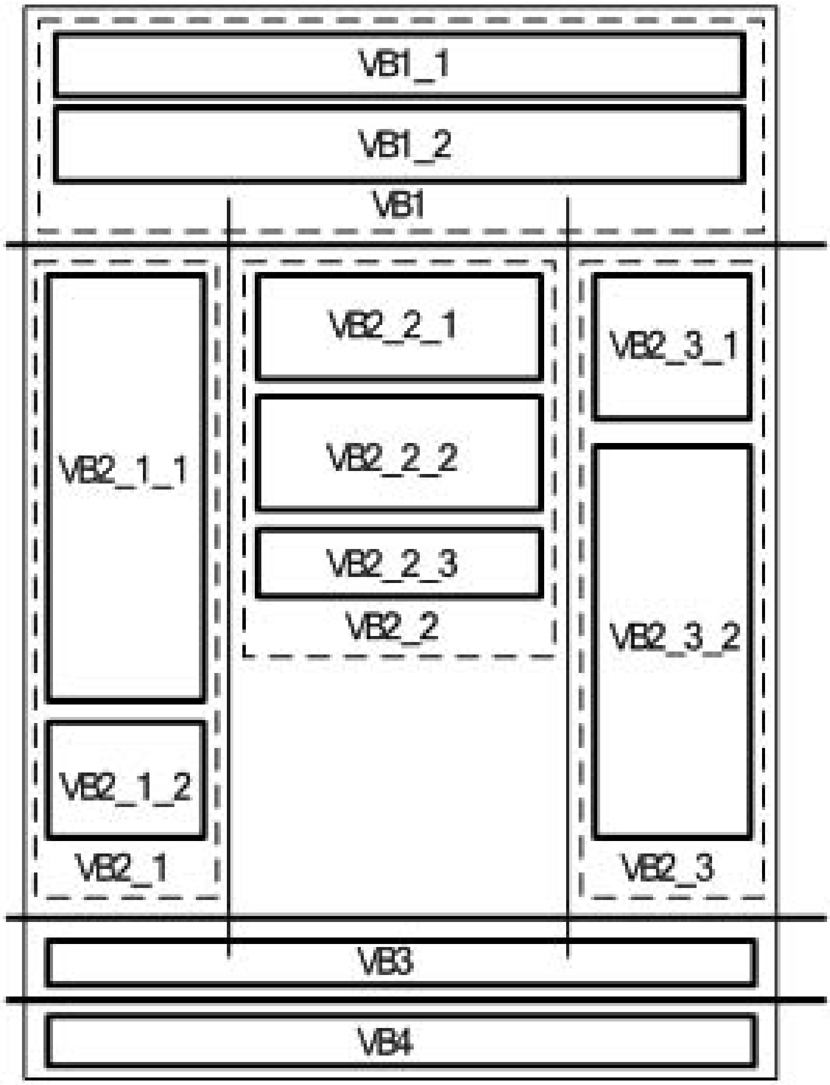


Figure 2.6: *Result of applying VIPS to Web page in Figure 2.5 [2]*

Authors conducted experiments on block retrieval to validate their work. These experiments start from retrieving ranked Web document where the rank here is for the overall page and not for a sub block inside it, and is referred to by Document Rank (DR). Next, page segmentation is applied to extract blocks inside these pages. This step produces the blocks set, representing all blocks in all pages. A Block Rank (BR) is then produced. Figure 2.6 shows an example of such blocks, BR determination is the process of assigning score to sub blocks inside the page, so that each page has many ranking scores, one for the whole page and many for these small blocks.Finally, the

system can choose one of three options to re-rank document.

1. considering the BR as the main and only way to re-rank document.

2. considering cases when multiple high ranked block occurs inside one page, so that the rank increases.

3. combine DR with BR using a special formula so that each of these ranks has a calculated influence on the overall rank. For example the rank for any Web page d is given by adding both DR and BR as in the equation 2.2

$$d_{rank} = \alpha.rank_{DR}(d) + (1 - \alpha).rank_{BR}(d) \tag{2.2}$$

   where d_{rank} is the calculated rank for a Web page d, and α is a number greater than 0 and less than 1, and is varied inside this range.

When using only the best block, results shows that performance is enhanced but it does not reach the level when using combination of page rank and block rank, especially when α exceeds 0.4.

2.1.5 Using Html Structures to improve retrieval

This technique, which was used in [11, 21, 2], targets simple text search and searching for images inside Html Web-pages. The technique is based on the vector space model, in which each digital image is represented by a vector of term weights. The weight of each term in the vector reflects its importance in identifying and classifying the document. The calculation of these weights depends on two factors:

- Term Frequency (tf): this represents the number of occurrences of a term (word) in a document.

- Document Frequency (df): this represents the number of documents in the collection containing this document.

A commonly used formula for weight is

$$\log\left(\frac{N}{df}\right) \cdot tf \tag{2.3}$$

where N is the number of documents in the collection, and $\log(\frac{N}{df})$ is called the Inverse Document Frequency or (idf). The authors of [11] created six classes based on Html tags. These classes were used to give more weight to terms occurring inside a Web document. The classes or tags are listed in Table 2.1.

Table 2.1: *The Six Html Classes*

Title	TITLE
Header	H1,H2,...,H6
List	DL, OL, UL
Strong	STRONG, B, EM, I, U
Anchor	A
Plain text	None of the above

The created index database is an inverted file. For each term T, the index database keeps the total number of Web pages that have term T, i.e, (df) and an inverted list. The inverted list is a sequence of pairs; The first element in each pair is the page ID, and the second element is the Term Frequency Vector (TFV). TFV contains the frequency of occurrence of T in that page for each class.

The term frequency vector TFV is tfv_1, tfv_2, tfv_3, tfv_4, tfv_5, and tfv_6, where tfv_1, tfv_2, tfv_3, tfv_4, tfv_5, and tfv_6 are the term frequencies of T in the Plain Text and they are Strong, List, Header, Anchor and Title classes, respectively. Indexing also employs stop words removal and stemming[4] of words. Then, it assigns the occurrence of the stem to a class and increments count, so each occurrence of a term in anchor description of document, d, is used to increment tfv_5 of the term in document d.

During indexing the crawler collects all the anchor text, to recursively index these links. The work also suggested the use of Class Importance Vector (CIV) (civ_1, civ_2,

[4] A word's stem is its most basic form. It represents a method used by the search spider to ensure that the greatest number of relevant matches are included in search results.

civ_3, civ_4,civ_5, civ_6), where civ_i is the importance factor assigned to class i. The search engine parses the query and uses the index built by the indexing engine to retrieve all Web pages which contain at least one of the queried terms. Then it uses the following formula to represents the importance of term T to document d.

$$w = (TFV \cdot CIV) \cdot idf \tag{2.4}$$

Finally, the search engine sorts the retrieved documents by non-increasing similarity and produces a ranked list of hyper-links to Web-pages, which the user can access. The authors used a genetic algorithm [25] to select a suitable value for CIV. To test the proposed work, they crawled all Web-pages that belonged to Binghamton University. Table 2.2 shows that the plain text class dominated other classes.

Table 2.2: *Term occurrence among six classes for the Binghamton University*

Class	Term	Percentage
Anchor Class	50,094	5.3 %
Header Class	46,314	4.9 %
List Class	22,684	2 .4%
Plain Text Class	723,060	76.5 %
Strong Class	91,684	9.7 %
Title Class	11,342	1.2 %

The approach finds the relevant set of documents by substituting the user query with a set of more general queries, then uses the search engine to expand and retrieve the set of images. This operation is monitored manually to determine the subset of relevant images. An image is relevant if it relates to the topic of the query.

2.1.6 AMORE: A World Wide Web image retrieval engine

AMORE [26] is a WWW image retrieval engine integrating several techniques to facilitate effective retrieval. This work assigns keywords to images to facilitate further retrieval of images. The system determines the semantics of an image based on the page containing it. This automatic operation is more difficult than manual assignment,

but manual assignment is not a feasible solution for large amount of collected data. Authors choose to parse the Html document and select terms near an image to associate it to terms. The problem is how to measure the nearness as a Html document can be structured in various ways. As a result, several heuristics techniques were required during document parsing and key word assignment as follows:

- Image file's name: Image file name is usually useful to determine the content of the file.
- ALT: The image field may contain a special ALT tag for alternate text, which is displayed if the image cannot be loaded.
- Image URL: The image URL may also be useful to identify an image.
- Html title: Because the title of the page gives the general idea around the overall page, it is useful to understand the semantic content of an image.
- Headings: Text appearing in Html headings in H1-6 tags are also useful. AMORE ignores some heading when it is not useful, as in the case if one heading is not connected to an image but to an alternative heading. This decision is carried on using Html parsers only, but in our proposed mechanism, the real physical distance is the arbitrator (see section 3).
- Other text in the page: Applying several heuristics to choose keywords within the page in order to associate terms to images.

2.1.7 Semantic Graphs for Image Search

This work does not deal with how we can search or index images, since it uses the already existent search engines to accomplish the searching and indexing. Rather, it suggests some related terms to begin the search with, which will improve search results by providing user with a suggested term that comes from Wikipedia after building a semantic graph of connected terms [3]. In this work, the authors proposed a new

Semantic based image search algorithm (SGIS). This system improve image search by utilizing collaborative knowledge from Wikipedia and network analysis to form semantic graphs for search-term suggestions. When a user types in a search term, SGIS automatically retrieves an interactive semantic graph of related terms that allow the users to easily find related images not limited to the supplied search term. Interactive semantic graph thus serves as an interface and a first stage to retrieve images through existing common search engines. This method significantly saves users time by avoiding multiple search keywords that are usually required in generic search engines. It also benefits users who do not possess large vocabulary and professionals who look for images on a regular basis; see Figure 2.7.

The authors compared SGIS with keyword image query systems. Thirty participants were invited to experiment with Google images search for a period of time. They were then asked to use SGIS to conduct search based on the same ad-hoc topics they previously put in, and then to answer survey questions on forms regarding novelty, flexibility and usability.

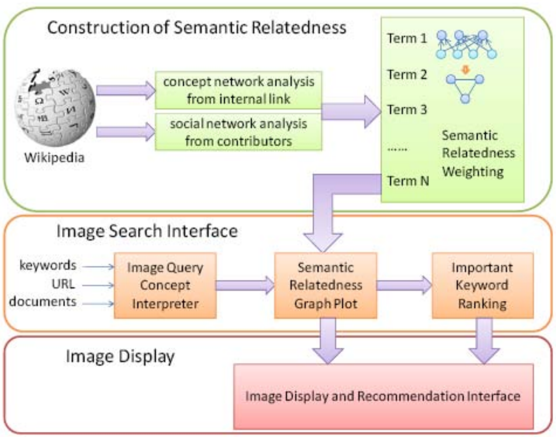


Figure 2.7: *SGIS semantic graph framework, adapted from [3]*

2.2 Content-based Image Retrieval (CBIR)

Many content based image retrieval systems can be conceptually described by the framework depicted in Figure 2.8 [4]. CBIR systems, just as TBIR, crawl Internet or local image database first. However, the difference lies on being interested on the image itself with its internal low level properties, in contrast to TBIR systems which concentrates on the textual environment that surrounds the image. CBIR shares some modules with TBIR: the crawler, and storing structure and scoring module. It is obvious that the internal implementation for these modules is different. Other CBIR modules shown in Figure 2.8 are explained below.

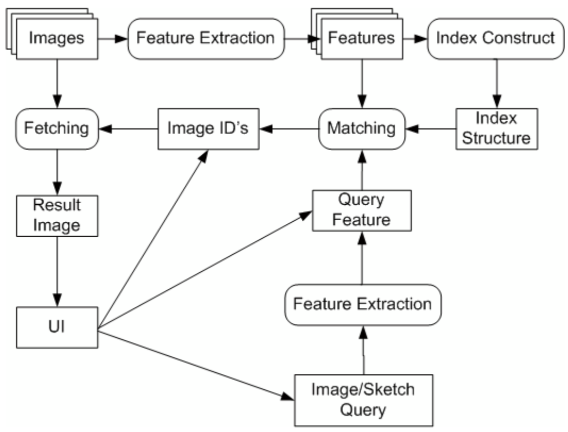


Figure 2.8: *General CBIR framework, adapted from [4]*

- Feature Extraction: this unit is responsible for finding the best set of features to describe the indexed image and to extract these features and feed them to other units when needed. Extracted feature are usually saved in a vector structure so each image is represented by one feature vector, which makes comparison between images (matching) more easier.

- Query Builder: this helps the system user to create the query of his search, the query can be created with different means. The user may select to browse the image collection and select a picture to find similar images by first extracting features in the feature extraction unit in the system. Alternatively, he may

supply a sketch or an image. Other ways also exist, such as submitting the feature vector explicitly to the system, e.g., submitting the color histogram in order to be matched with the already saved features.

- Image Matching: after providing the query to the system, the system has to extract features from the query and apply a matching function, which results in a set of candidate resultant images sorted by the degree of similarity to query features. The system then fetches these result from the database collection and provides the user with a suitable user interface to view results.

Content based image retrieval systems are more convenient when applied on a closed domain to search inside because images will have something in common which allow to specify a set of discrimination features for matching more easier. An example of a closed domain is searching within a hospital medical image database. On the other hand, the performance of CBIR system will sharply degrade when applied to an open and diverse types of image collections it is extremely difficult to choose a suitable set of features for classification. While different images may share the same features, using a histogram as a classification of features will face a problem because the histogram based technique is known to be sensitive to small color variations caused by lighting conditions [27]. So, irrelevant images may share the same histogram, and other images containing identical objects may have different histogram. As an example Figure 2.9 contains two images for a turtle with different values for histogram.

What applies for histogram applies also for other features, such as texture, patterns and boundary. If we take the boundary feature as an example, it is obvious that different types of cars have different boundary lengths, especially if different models and dimensions are considered. The retrieval based on colors usually yields images with similar colors. Retrieval on texture does not always yield images that have clearly the same texture unless the database contains many images with a dominant texture [4].

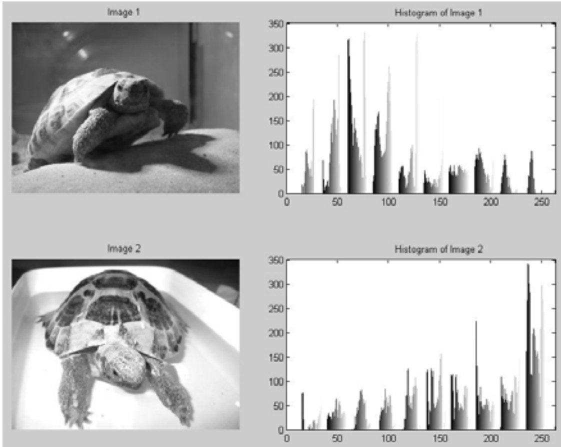


Figure 2.9: *Two Images with their histograms*

All such reasons make searching images using their contents a non-semantic approach, and so many researchers used it only to support text-based approach as in [5], or alone for closed domain usage.

2.2.1 Context-Based Image Retrieval on the WEB (COBWEB)

COBWEB retrieves images based on the content of the images (visual features) [5]. The system provides a solution to the problem of automatically storing and retrieving images from large image databases over the Internet. It encapsulates algorithms for the automatic storing of the visual features of the images that can be used as a means for retrieving similar images. This system also provides a straightforward, cross-platform (Java based) user interface that helps the user perform queries and image retrieval. Also, the system provides a robust communication subsystem that guarantees satisfactory retrieval times; see Figure 2.10. Figure 2.10 explains the interaction between the user who instantiate the search and the COBWEB system, user make use of the search engine intelligence to navigate and query the image database.

Interactions between the components of the proposed system are shown below in Figure 2.11.

The scenario of usage start by submitting a sample image or sketch to the search engine, this image is converted first into a feature vector using the Feature Extraction

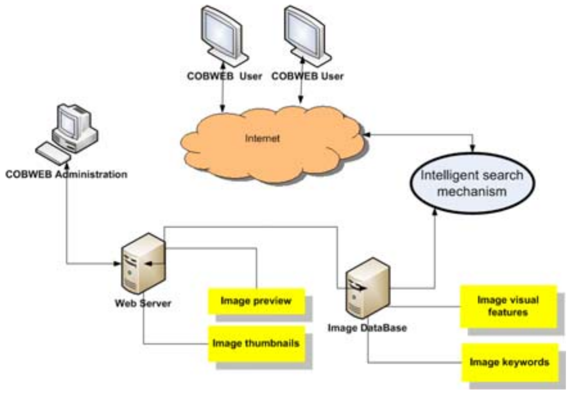


Figure 2.10: *COBWEB architecture, adopted from [5]*

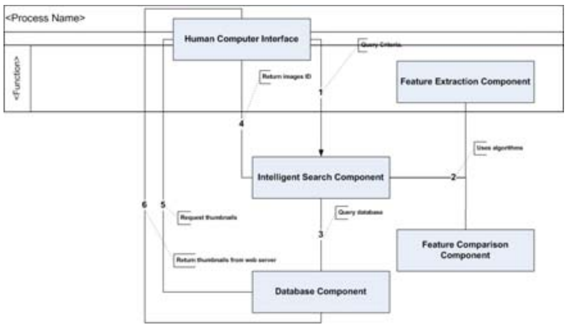


Figure 2.11: *Interaction between system components*

Component (FEC). At this time the Intelligent Search Component (ISC) can operate on this vector, and try to find a similar feature vector satisfying the searched query on the database; the comparison is carried on inside the Feature Comparison Component (FCC). After finding some matching the ISC return the ranked result to the Human Computer Interface (HCI), which in its turn displays the images thumbnails to the user, the ISC serves as an intermediate tier between the HCI client and the database. ISC power lies on the use of effective and efficient mathematical algorithms, capable of extracting a number of well defined features (visual features) from images of arbitrary size. Proposed algorithms integrated in the first prototype of the COBWEB system are:

- Global color histogram.
- Local color layout

- Large scale color texture.

Finally, it is worth to mention that COBWEB project does not provide any assessment methodology to validate its correctness.

2.3 Incorporating distance as a factor in image search

Different techniques have been developed to solve the problem of tagging and assigning keywords to images on the Web. However, the main problem of these techniques is the huge number of on-line pages that need to be annotated and processed to comply with these techniques requirements, and this expresses the need for a technique with no prerequisite conditions. In our thesis we exploit features available by TBIR technique, TBIR is usually a prerequisite step for using CBIR as the later one uses an existing image as the query, and this image can be found using TBIR which uses only a term to look for an image. Many other techniques, e.g., [2, 11, 28], use the distance as one of the most important factors to determine the degree of relevancy between a query term (word) and an image inside any single Web document. Existent proposals simulates the measurement of this distance and use any of the following:

1. number of intervening words between the text and the image ($numwords$).
2. number of intervening full stops such as ., ?, !. ($numfullstops$).
3. number of other sentence-ending punctuation between the text and the image.
4. number of intervening table data tags $< td >$ ($numTDs$) and table rows $< tr >$ ($numTRs$) between the text and the image.

The authors in [28] used the following simple equation 2.5 to calculate the distance between an image and text in search query:

$$Distance = (1 \cdot numwords) + (10 \cdot numfullstops) + (5 \cdot numTDs) - (10 \cdot numTRs) \quad (2.5)$$

where the multiplication factors 1, 10, 5 and 10 are the corresponding weights. Naturally, this is logical as intervening $numTRs$ for example means a new row, which requiring increasing distance (weight).

Figure 2.12 shows the picture of Obama while signing the order to close Guantanamo. This image has a headline beside it with the text "Obama signs order to close Guantanamo in a year(AP)". Simple processing to this line will result in tagging the image that it may relate to Obama. This is valid if this line is located near the image inside Html code but it is not the case always as sometimes there is a huge gap between the image location and the headline. Web designers choose to use JavaScript to fill the headline from a far location inside the document textual source, and so a simple processing of this code will never relate this image to Obama.

The major problem that makes these techniques fail to compute a correct relevancy and assignment of stems from different reasons, e.g., the dependency on Html source (text) structure to make decision, which usually results in wrong weighting of words, see Figure 2.13.
The error in weighting originated from many scenarios, e.g., ordering images in tabular mode or using JavaScript to fill in some areas at rendering time. Figure 2.14 shows how the textual treating of the Web pages may result in erroneous keyword assignment[5].

In [11], the authors calculated the weight of each word (term) based on its location, or on the distance from images: near surrounding words relate to the image more than words far from the image. This can be erroneous as these words may exist near the images in the source file, but when rendered with any Web browser, e.g., Internet explorer (IE), or FireFox, they are far away from the image.

Our proposed approach, explained in chapter 3 will address the shortcoming and problems of previous heuristics for incorporating distance as an important factor in image search.

[5]http://www.archinect.com/gallery/

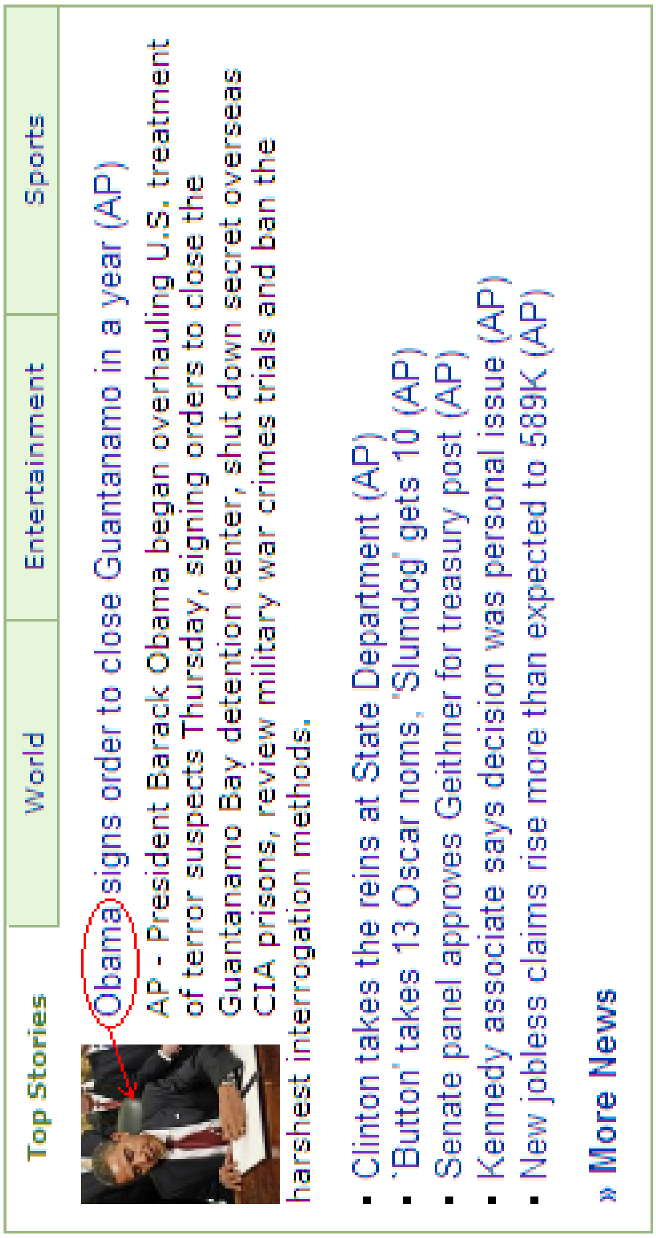


Figure 2.12: *Example of an image with its related description text.*

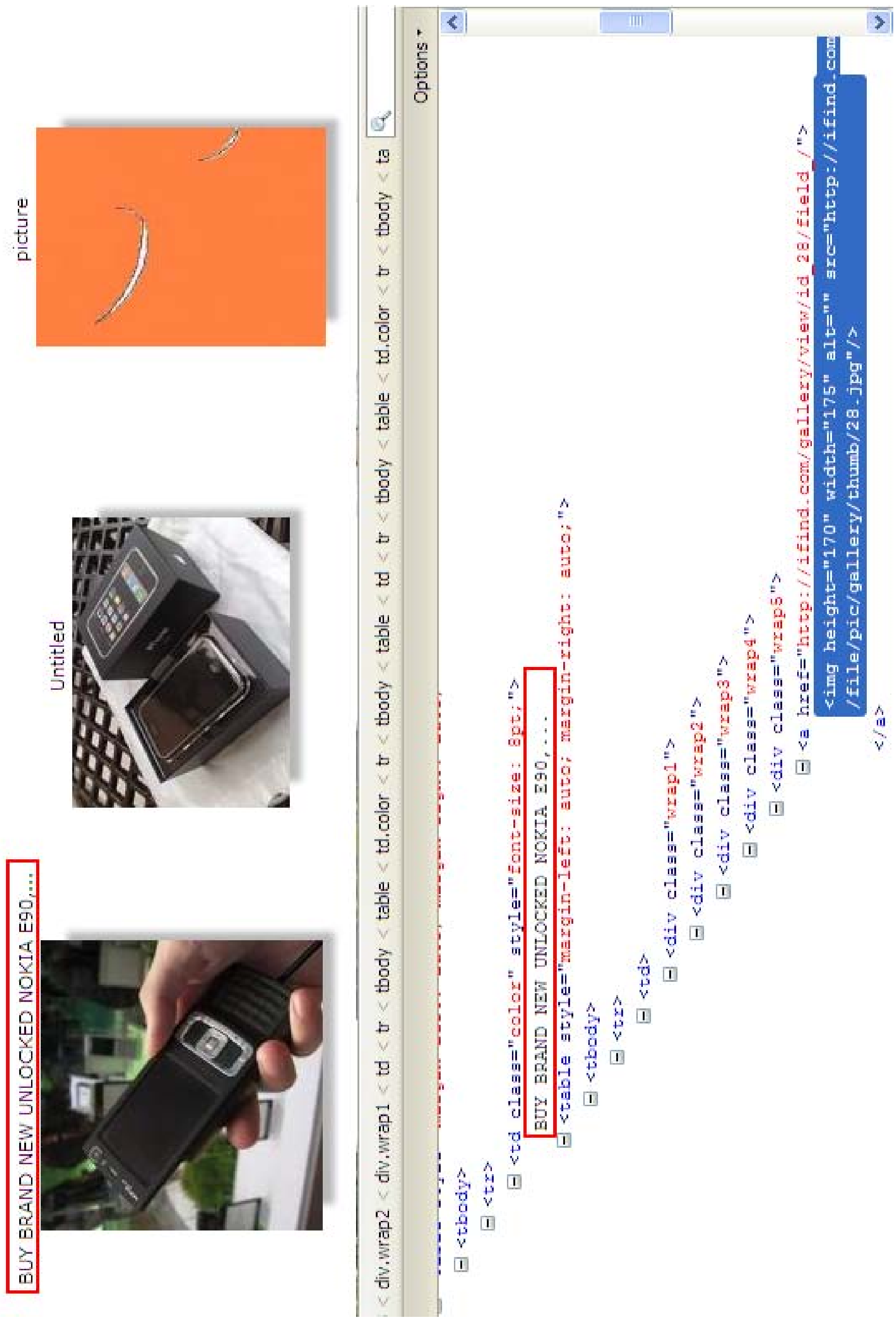


Figure 2.13: *Example of erroneous text relevancy calculation*

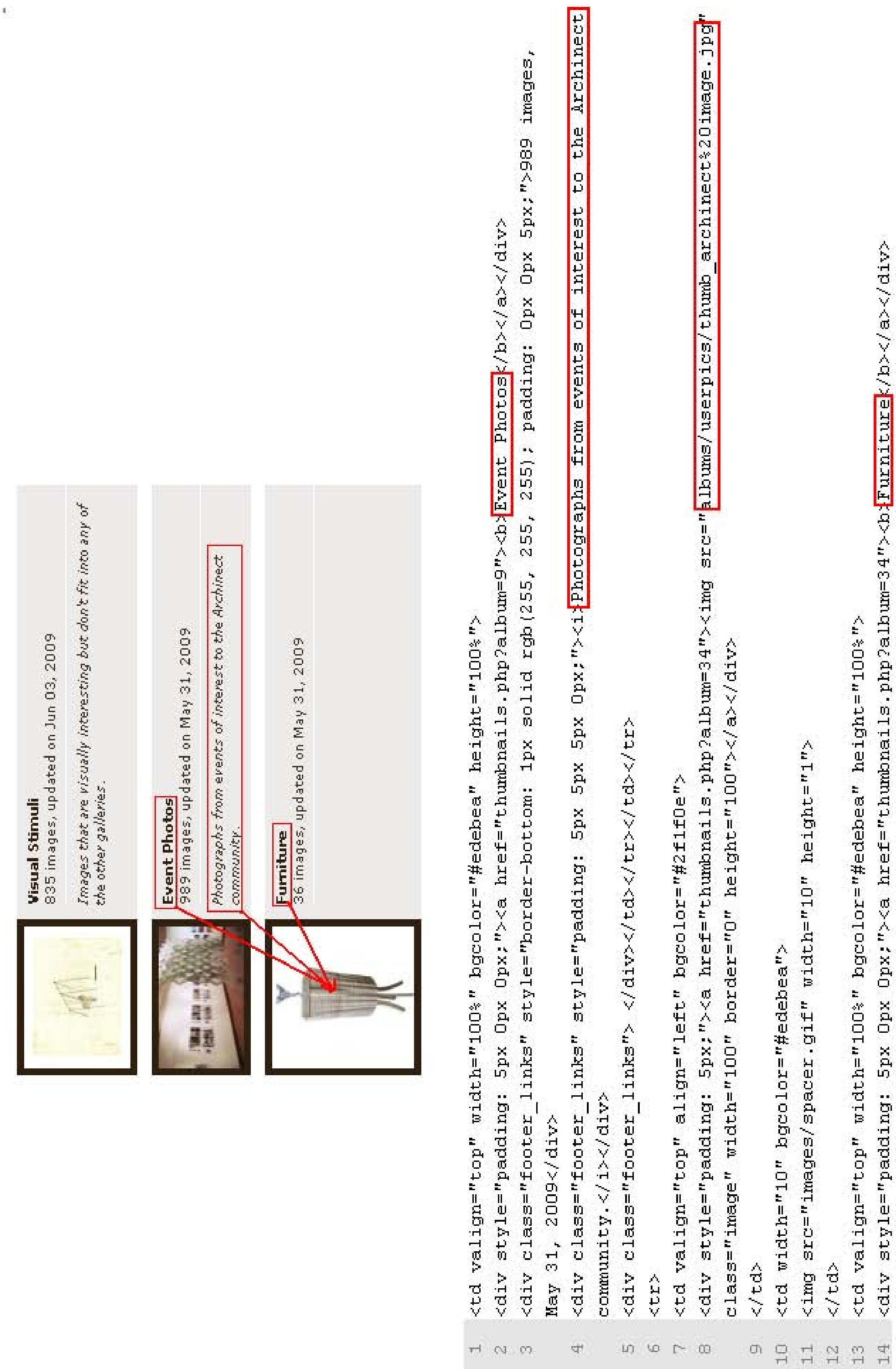


Figure 2.14: *Example of erroneous text relevancy calculation, tabular problem*

Chapter 3

Employing Physical Distance In Re-ranking Image-Search Result

Text within a search query is associated more strongly to an image inside the same document if the term is closer to an image than any other images in that page [16, 17]. In this chapter, we present a novel method for image re-ranking based on a true physical distance measurement between the image and the queried keyword. Physical distance is the distance represented by a straight line between two locations on the web-page in pixels after completely rendering the web-page.

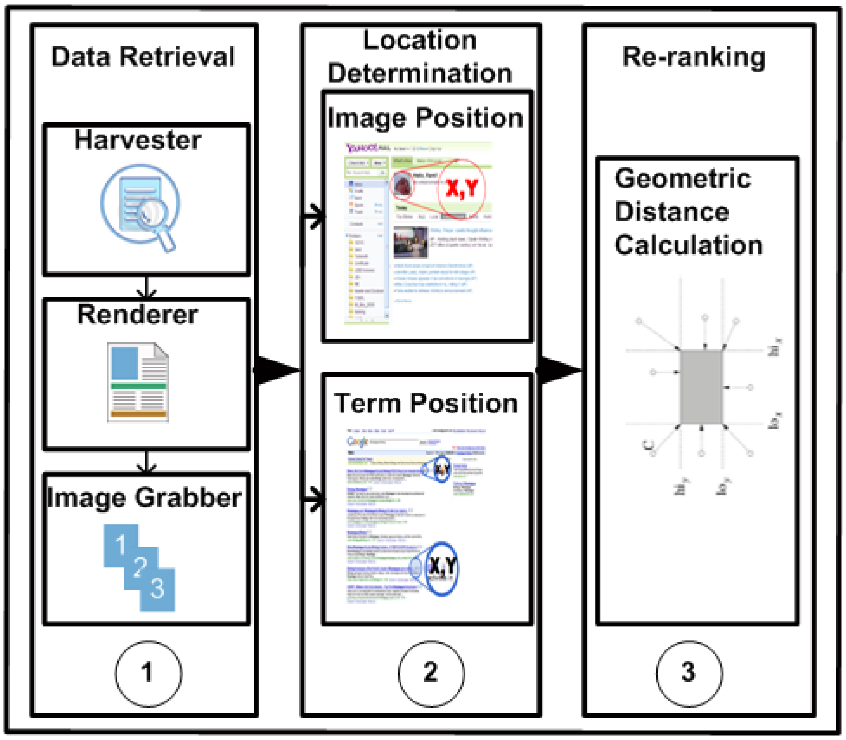


Figure 3.1: *True distance Based Image Search Architecture*

This method ensures obtaining the best result since it measures the exact distance in pixels. Figure 3.1 presents our system's high level architecture. The system consists of three main modules:

- Data Retrieval Module: performs the initial data harvesting and preparing the data set that other modules will be operating on.
- Location Determination Module: this is responsible for asserting image location as well as query keyword location.
- Re-ranking Module: it applies re-ranking algorithm based on the physical distance.

3.1 Data Retrieval Module

The Data Retrieval Module is responsible for gathering required data from the Web, and preparing suitable input for the next processing steps, see Figure 3.2.

Data Retrieval Module receives a keyword as input, and generates two distinguishable outputs, the first one is the WP_C, which is a snapshot of the whole Html page as if it is opened and viewed by any Internet browser, and the second output is RP_{URL}, which represents the search resulted image. This generation encapsulates different processing unites: data gathering and harvesting, image grabbing and web pages rendering (capturing), a detailed explanation of these processes follows in the next sections.

3.1.1 Harvester

This module accepts search query terms and then collects corresponding results from a common image search engine . The system obtains two outputs from the search engine and persists them to a database as a preparation step for the rendering phase. The outputs returned from search engine are:

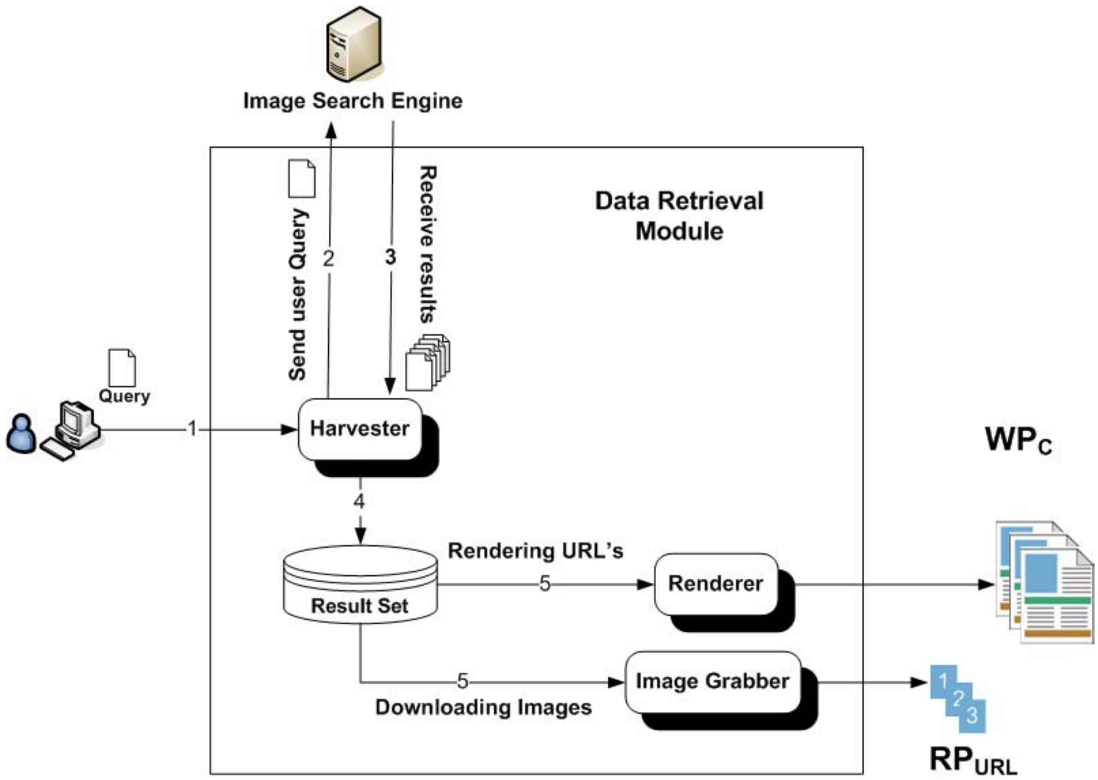


Figure 3.2: *Overview of Data Retrieval Module*

- The URL of the web-page containing a resulted picture (WP_{URL}): when search criteria happens to match an image, the search engine provides a link to the web page containing this image.
- Result Picture source's URL: this is the path to access the resulted image on the Internet.

The above two outputs extracted from the quired search engine are consumed in the next processing step, e.g., the WP_{URL} will be used in the renderer step, and the result picture source URL will be used in the image grabber step to produce the RP_{URL}.

3.1.2 Renderer

The renderer module is responsible for completely rendering and capturing a snapshot of each WP_{URL} creating a complete web page image WP_C. The WP_C is a snapshot of the whole Html page as if it is opened and viewed by any Internet browser. We choose

to render web-pages using a rendering resolution of (1024×768). So, for each WP_{URL} in the result set comes after querying a common search engine, different processing takes place as follows. Rendering a web-page needs two thing to accomplish the task, firstly, we need the web-page textual source, and secondly, a software agent to draw the source files. This module request for some resources on the Internet, which is here the WP_{URL}, the response is usually a raw Html representing the page layout. In order to generate an image of the web-page under processing, we first need to draw it. Drawing or rendering web-page is the same process done by any web browser when we uses it to request any web-page on the Internet. Our system draws the image on a panel on top of operating system desktop, after drawing the WP_{URL} we capture the active form (panel) on the desktop and save the generated image. Sometimes the renderer encountered a long web-page that cannot be rendered once on the screen (its height greater than desktop height), so the renderer shall detect this and keeps scrolling the web-page and capturing the newly appeared portion of the page and save images (sub image). This is repeated until the renderer reaches the end of the web-page, after that renderer collects the sub images generated while scrolling and combine them into a one whole image represents WP_C. Our system needs an accurate WP_C that really represents the corresponding web-page. So, we use an underlying API to use renderer of a well know browser on different platforms, on other words, Html renderer engine of Internet explorer will be used on top of Microsoft Windows platform, Mozilla will be used on top of Linux and Safari on Mac [29].

After Renderer step, the system has an image representing the whole web-page for each WP_{URL} found in the database. This image file is saved to the local disk and a corresponding database record is updated to reflect new changes. Renderer exploits the Standard Widget Toolkit (SWT) [29], which is an open-source embeddable Java Kit. It is used in the development of a wide range of Java SWT standalone and Rich Client Platform (RCP) applications.

3.1.3 Image Grabber

In parallel to the rendering step, image grabber downloads the RP_{URL}. The downloaded images undergo some modifications, such as, resizing and format conversion.

Diverse web-pages are parsed by our system, these pages includes non homogeneous images both in size and format. For example, images may be accompanied with some attributes that force it to be rendered in the browser with different sizes than the original size. Width, height and style are examples of valid attributes for img tag[1] in Html, these attributes specifies the rendering size of the image in the Web browser; the Code Snippet 1 represents an Html img tag for an image with the name *laptop*, note that this image is saved on the server local disk inside a directory called *images*, this image have a size of 400 by 250, but the web designer choses as Code Snippet 1 indicates to make the size less than the actual size by providing new values for width and height, these attributes will instruct the browser to display the *laptop* with a smaller size.

Code Snippet 1 IMG tag attribuites

```
<img src="images/laptop.jpg" width="132" height="170">
```

Without resizing RP_{URL} appropriately, determination of the exact location of RP_{URL} inside $WP_{C)}$ would be difficult, because we use our software system to recognizes the location of this images RP_{URL}, and without giving the software an accurate and exact input it will fail to locate this image inside WP_C as the software cannot handle variation in size, this explain why sub-image RP_{URL} needs to be identical to how it appears on the browser. Image grabber module extract image's width and height from the image tag in the source file using an Html parser, these values are used to determine whether the image size is overridden or not, and so resizing whether is required or not. Resizing is carried out using one of the simplest scaling algorithm, it scales an existing image by first sampling pixels in the source image to produce the new scaled image by replicating rows and columns of pixels to scale up or omitting (deleting) rows

[1]http://www.w3schools.com/tags/

and columns of pixels to scale down [30]. Image type conversion also applied for every RP_{URL} we download. Images on the web have different type format, e.g., PNG, JPG, Bmp, GIF, we choose to unify all the downloaded images to one format which is *JPG*, in order to make all processed images inside the system deal with the same file type.

3.2 Location Determination Module

The main aim of this module is to locate both the coordinate of RP_{URL}, and the searched-term inside WP_C; see Figure 3.3, the result of this step is recorded to database in order to be used for the re-ranking module. The extracted coordinates for the RP_{URL} represents two points on the plain, the first one represents the x and y of the bottom left corner of the image, and the second point represents the x and y of the top right corner of the image. The extracted coordinates for the keyword (searched-term) is a single point represent the center of the term.

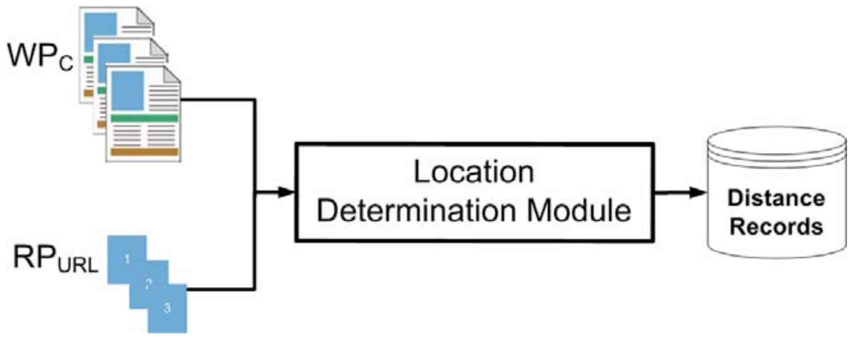
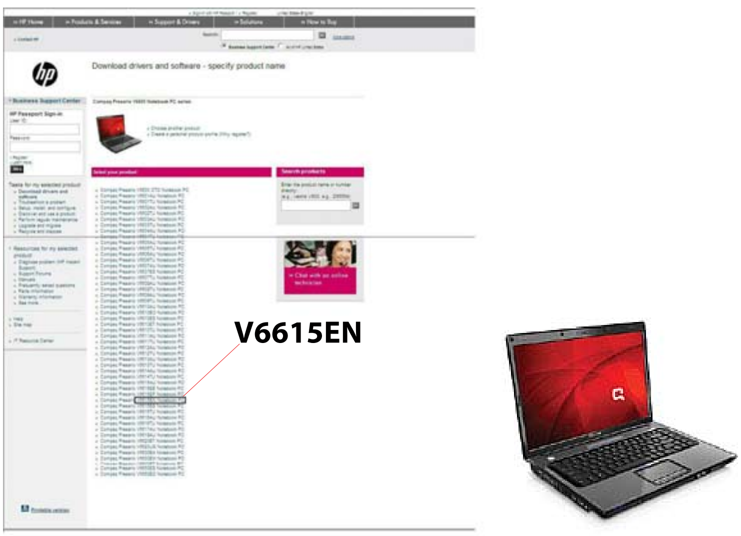


Figure 3.3: *Location Determination Module*

3.2.1 RP_{URL} Location Extraction

The location of each RP_{URL} image inside the corresponding WP_C image is determined by applying image processing algorithm on WP_C and RP_{URL} images to find the exact location of the sub-image inside the full image. To accomplish this task we uses different image processing techniques like, pattern recognition, correlation, morphological operations and template matching.

Algorithm 1 shows pseudo code of our algorithm for locating RP_{URL}, this code starts by reading two images, the first one is the WP_C, and the second is the RP_{URL}. In some cases images returned from search engine may contain some old results (crawled long time ago), old result usually contain removed or moved resources, this may affect the Image grabber operation making it fails in downloading the RP_{URL} because of unavailability of requested resource on the web server, our system will treat these images as least relevant images (assign them a maximum value of distance). Template matching technique uses these two extracted images, it calls the WP_C *target* image, and RP_{URL} *template* image. Figure 3.4 shows an example of *template* image and *target* image.



(a) *Target WP_C* ***(b)*** *Template RP_{URL}*

Figure 3.4: *An example of target and template images*

After reading these images we convert them into gray level images, the conversion process averages the image pixel's value. Each pixel in the colored image consist of thee values (R, G and B) stands for Red, Green and Blue, its values are between 0 and 255. Averaging (R, G, B) values and setting the resulted value of all R, G and B, e.g., if we have a pixel with the values (56, 200, 134) for (R, G, B) then after averaging the new generated (R, G, B) will be (130, 130, 130) this reduces the complexity of dealing

Algorithm 1 Locating RP_{URL} in WP_C

```
(1) Read both the Template and Target Images
(2) If Template image is missing then go to step (7)
(3) Convert images in step (1) into gray level images
(4) Apply Sobel edge detection algorithm on gray level images from (3)
(5) Apply template matching algorithm on the result of step (4)
(6) Map the results from (5) to location of template image inside the -
    target image and go to step (8)
(7) Assign this image with the farthest location coordinates, and tag it
to be assigned a maximum distance.
(8) Return the calculated location.
```

with three values per each pixel. After generating gray level images we use them to create binary images of only black and white colors (pixel values of 1's and 0's only), this is done by applying some morphological operations to detect edges inside these images. Edges are areas with strong intensity contrasts where we can find a jump in intensity (pixel's values) from one pixel to the next. Edge detecting is the process of finding these intensities inside an image.

There are many edge detection algorithms, but in our work we use the Sobel operator [30]. The Sobel edge detector uses a pair of 3x3 convolution masks, one for finding the gradient in the x-direction (columns) using the following kernels(matrices):

$$G_x = \begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix}$$

and the second matrix estimates the gradient in the y-direction (rows) and uses the following kernel:

$$G_y = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

G_x and G_y are two small images (matrix) which at each location contain the horizontal and vertical derivative approximations, respectively. The convolution mask (G_x, G_y)

is usually much smaller than the actual image (*template*, *target*). So, the mask is slide over the image, manipulating a square of pixels (dimension of the mask) at a time. Figure 3.5 shows the result of applying Sobel edge detection to images in Figure 3.4. To obtain the combined magnitude we use equation 3.1.

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

So, to apply Sobel operator on image I to extract the edges for the x-direction (G_x), we use the following equation 3.2:

$$I(x, y) = \sum_{s=0}^{m} \sum_{t=0}^{n} G_x(s, t) \cdot I(x + s, y + t). \tag{3.2}$$

where m and n are the dimensions of Sobel kernel, I is the images on which we apply Sobel operator.

Next, the algorithm continues by filtering the resulted edge image of the *target* with the edge image of *template* to find the correlation between the two images. Correlation is a robust and general technique for pattern recognition [31] which proves a good results in our case.

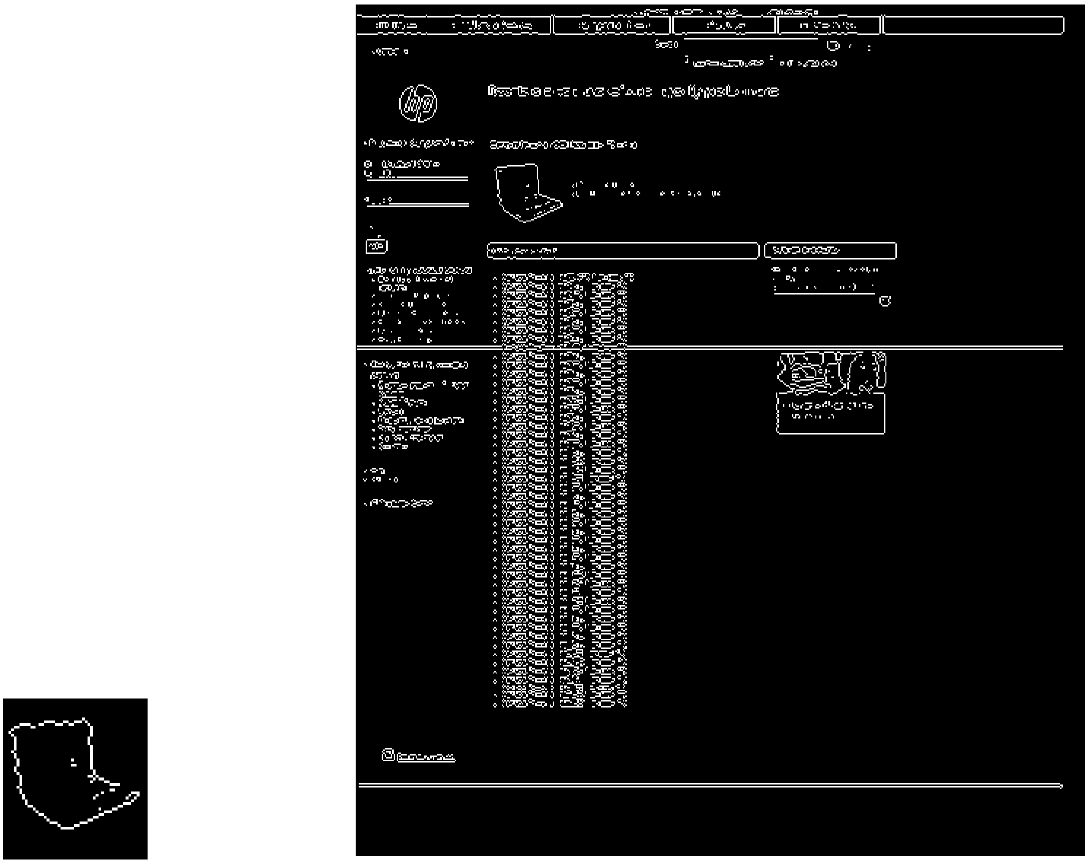
The matching process slides the center of *template* image (smaller) over each (x, y) point in the *target* image (larger) and calculate the sum of products between the corresponding pixel values of both *target* and *template* images, this is carried on over the whole area of the template, note that the position with the highest score calculated is the position of the *template* image (RP_{URL}). Therefor, we search for the maximum score and extract the bottom left and the upper right corners of the *template* image; for example Figure 3.6a shows the result of applying template matching on the *template* image in Figure 3.4b and *target* image in Figure 3.4a. Figure 3.6a shows a peak value (Maximum Score) at the point that represent the location of the *template* image as Figure 3.6b also shows. Figures 3.7 and 3.8 shows another example of applying template matching to locate the *template* image inside the *target* image.

The correlation between *template* and *target* images is given by equation 3.3

$$correlation = \frac{\sum_{i=1}^{N} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 \cdot y_i^2}} \tag{3.3}$$

where x represent the *template* and y represents the *target*, N is the dimension of the *template* image, we assume here a square image with height equals N.

After this step we get the location of *template* image inside the *target* image, we

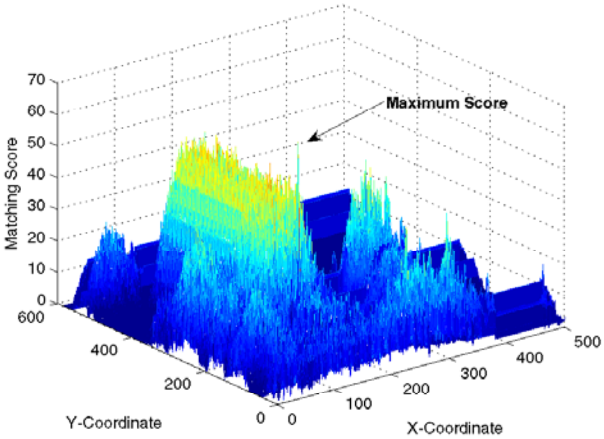


(a) *Target after applying edge detection*

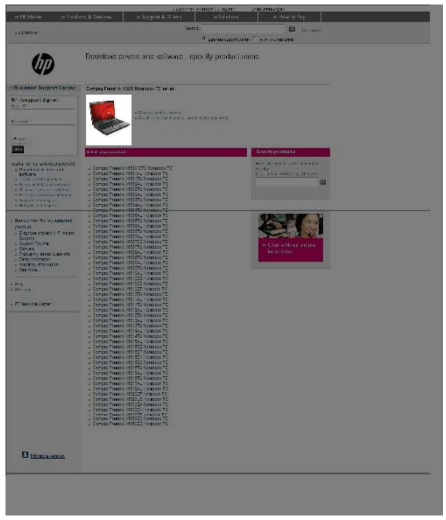
(b) *Template after applying edge detection*

Figure 3.5: *Target and Template images after applying Sobel edge detection*

get $R_{lo} = (lo_x,lo_y)$ coordinate represents the RP_{URL} bottom left corner X and Y pixel values, and $R_{hi} = (hi_x,hi_y)$ coordinate represents the RP_{URL} upper right corner X and Y pixel values.



(a) *Result of applying template matching*



(b) *Mapping template matching to location*

Figure 3.6: *Result of Matching and mapping of template image to location*

3.2.2 Query term location extraction

This step determines the location of the searched-term inside the WP_C. This is the final step of processing before measuring the physical distance between the queried term and the sub-image (RP_{URL}). The process encapsulates processing of the web-page Document Object Model (DOM) tree, DOM specification published by the World Wide Web Consortium (W3C), represents a new advancement in handling structured documents mainly, XML (eXtensible Markup Language) and HTML [32].

DOM provide programmers with a rich Application programming Interface (API) that help them constructing their code in an organized form, as all the produced code will adhere to the DOM specification, which ensure compatibility across different DOM implementation. DOM facilitate accessing and manipulating stored information inside

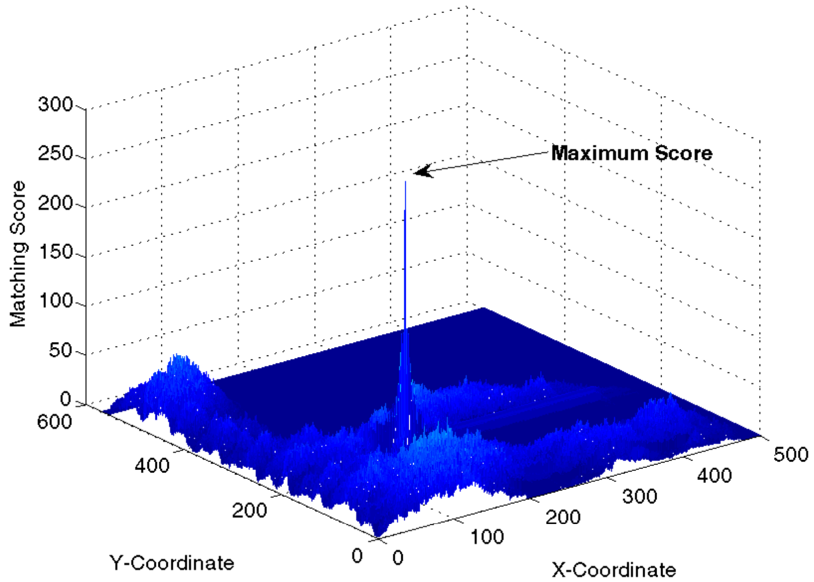


Figure 3.7: *Result of applying template matching*

structured document (Html). DOM documents that represent an Html documents constructed as a tree structure, the tree is constructed also by a set of nodes, nodes also have child nodes and so on, consider the Html Code 2.

Code Snippet 2 Basic Html Document

```
<HTML>
<HEAD>
<TITLE>This is an example</TITLE>
</HEAD>
<BODY>
<B>Some Bold Text</B>
<IMG src="..."/>
</BODY>
</HTML>
```

Html Code 2 shows the tree structure of a simple document, it shows the parent-child relation between tags. At the top most (root) of the document we found $< HTML >$ tag, underneath it, the other tags become children of the root tag. These child tags can also have their children; for example, $< B >$ tag is a child node for the $< BODY >$ tag, notice that $< BODY >$, in turn, is a child for the root $< HTML >$

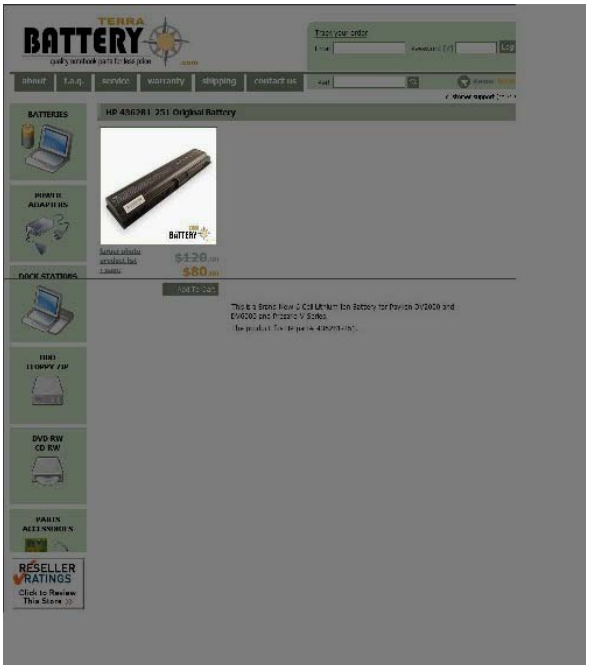


Figure 3.8: *Mapping template matching to location*

tag. In this step we execute a JavaScript function that manipulate the DOM tree, with the outline shown in Algorithm 2. This code is responsible for locating each occurrence of the searched term inside a single web-page by executing this script within an already opened web-page. Web-browsers provide users with a simple search (find) utility for words inside an opened web-page, sometimes web-browsers also highlighted the result of the search, our determination module make use of this utility to find occurrences of the searched term, and then determine the exact location inside the web-page.

Algorithm 2 Function to get a specific term locations within a web-page

```
(1) Parse the web-page and construct DOM tree.
(2) Highlight term occurrences using built-in find utility.
(3) Tag the node with term matching from previous step
(4) Create object ranges of the nodes from step (3).
(5) For each selected node extract the x and y coordinates and store them,
if no node exist, tag the corresponding image to be assigned a maximum
distance.
```

While we collect the coordinates of the queried term in the web-pages, we traverse the DOM tree from the term we look for and collect the nested parent tags for that term, for example if the search term is v6615en, we traverse the Html DOM tree from the occurrence of this term until reaching the Html Root element, and for each node we record the type of this tag, see Table 3.1. Note that from Table 3.1 we can incorporate further processing and ranking based on the degree of importance of the parents tag of the queried term, like in [11].

Table 3.1: *Parent tags for term v6615en*

Term	X-Axis	Y-Axis	Parent-Tag
v6615en	915	353	A,TD,TR,TBODY

3.3 Re-ranking Module

The goal of the Re-ranking Module is to re-rank results from initial processing steps of Data Retrieval and Location Determination Module. It is the final step after which the new ordering of search result is available. The inputs for this step are both the location of terms inside the page and the rectangle coordinates of the RP_{URL}.

3.3.1 Computing Physical Distance

After both the query term location and image location is determined, the distance between every image and query word is calculated. The distance is the shortest path measured from the center of a term to the nearest edge of the image rectangle, and not the center of the image, using this measurement criteria assured that all images with different sizes will be treated equally without biasing smaller images, consider the case that the term is touching the bottom edge of the image, if we choose to measure the distance from the term to the center of the image, so the distance will be varying with the variation of the image size, in contrast to measuring the distance from term to the nearest edge which will produce always the same value. The process of finding

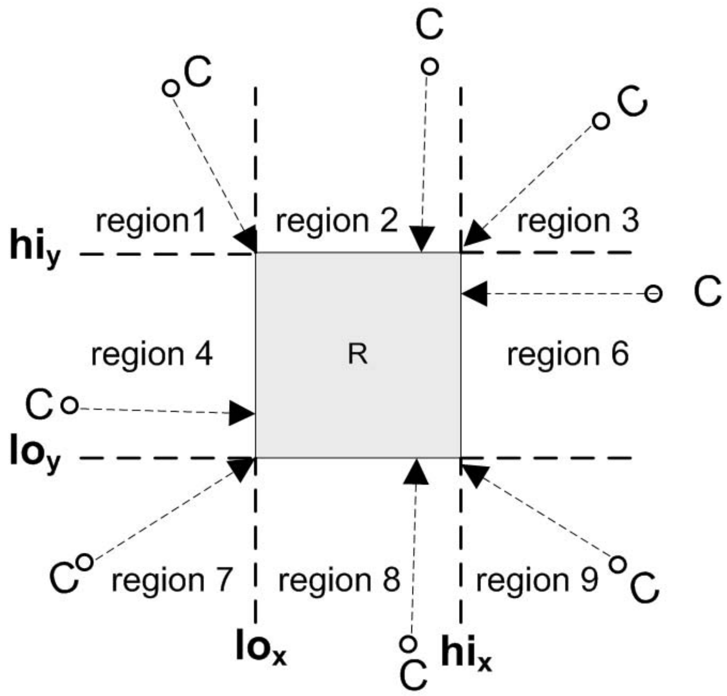


Figure 3.9: *Measuring Distance from point C to Rectangle R, adapted from [6]*

the distance between the center of the word inside the page and the closest edge of the image is a typical problem involving simple geometric primitives [6]. Consider the location of the term as the point (C), and the image (RP_{URL}) as a rectangle (R); Figure 3.9 is used to determine the region of the term C from the available nine regions, this whole image represents the WP_C and the term is represented by the point C and the RP_{URL} is provided as the rectangle R, on the other hand, Figure 3.10 shows the corresponding square distance for different location (regions) of term C, this distance is calculated to the closest edge of R (RP_{URL}). R is defined by its lower left (R_{lo} = (lo_x,lo_y)) and upper right (R_{hi} = (hi_x,hi_y)) corner points.

It is mandatory to determine exactly the region that the term happened to be inside, we apply Algorithm 3 to find the distance based on the region. At first we check weather the x-coordinate of the term is to the left or to the right of the image (R). if so we calculate the distance using the distance square law square($lo_x - C_x$) or ($C_x - hi_x$) based on the location of the term, to the left or right, if the x-coordinate of the term is not to the left nor to the right, then the x-direction distance is zero. We apply

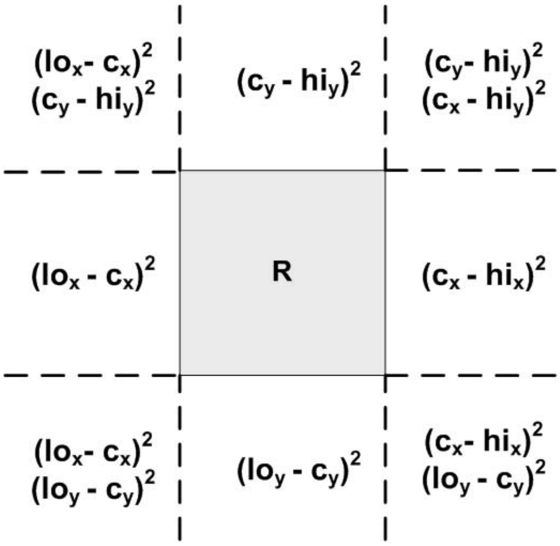


Figure 3.10: *Distance contribution per each region, adapted from [6]*

the same thing to the y-coordinate of the term C, but this time checking weather the y-coordinate is above or bottom of the image R, if so we calculate the distance using the formula $C_y - hi_y$ or $(C_x - hi_x)$, we add the distance to the one calculated in the previous step.

Algorithm 3 Calculating Distance between term C and rectangle R (RP_{URL}) image

distance = 0
if($C_x < lo_x$)
distance += square($lo_x - C_x$)
else if($C_x > hi_x$)
distance += square($C_x - hi_x$)
if($C_y < lo_y$)
distance += square($lo_y - C_y$)
else if($C_y > hi_y$)
distance += square($C_y - hi_y$)

Calculation of distance is carried on by determining the coordinate of C, and then applying the corresponding squared distance law shown in Figure 3.10, e.g., if the term is located in the region 9 in Figure 3.9, then we have to apply the equation 3.4.

$$D = (c_x - hi_x)^2 + (lo_y - c_y)^2 \tag{3.4}$$

If we return to Figure 3.6b we can notice the location of term highlighted near the bottom of the page, the location of this term to the laptop image lies on the region 9, so the equation 3.4 is also valid for this case to calculate distance D, Table 3.2 shows the corresponding square distance for each available region as in Figure 3.10.

Table 3.2: *Squared Distance for each region*

Region	Used Equation
1	$(lo_x - c_x)^2 + (c_y - hi_y)^2$
2	$(c_y - hi_y)^2$
3	$(c_x - hi_y)^2 + (c_y - hi_y)^2$
4	$(lo_x - c_x)^2$
5	0
6	$(c_x - hi_x)^2$
7	$(lo_x - c_x)^2 + (lo_y - c_y)^2$
8	$(lo_y - c_y)^2$
9	$(c_x - hi_x)^2 + (lo_y - c_y)^2$

3.3.2 Re-ranking Based on Physical Distance

After calculating the distances that separate terms and RP_{URL}'s, we can now apply our ordering algorithm, which orders the result using calculated distances. In order to tune our algorithm, we add to our ranking varying portion of the external ranking comes from the selected search engine ranking.

Our algorithm uses a parameter (α) to vary the amount of contribution of external ranking into our ranking. Re-ranking algorithm varies α between 0 and 1.0, with a period of 0.2, based on the equation 4.1.

$$C_I = \alpha \cdot O_I + (1 - \alpha) \cdot E_I, \tag{3.5}$$

where E_I is the External rank for image I, and O_I is our ranking for the same image I. Then the new combined composite ranking is C_I. When α is one, then we completely neglect External ranking (our pure ranking), and when it is zero, the ranking is pure Externals, this mean omitting our ranking. A variation between the two extremes

(from zero to one), varies the contribution of O_I and E_I on the final ranking C_I.

In order to incorporate two different ranking from to different sources we have to unify the scores unites which comes from these two sources. So, we choose to score our result as based on the follows fact:

- The minimum the distance between a term and an image, the more relevant that term is to that image in pixels.
- To use the distance as a score we have to normalize it, so that it fills the period from 0 to 1.0, or in percentage from 0 to 100.
- For each record representing one term, there is may be one or more occurrences of the term in that web-page, our system collects all these occurrences, but we care only about the minimum distance, still the mean, average and maximum are available.

We use the term distance as our ranking main factor, in other words, the shorter the distance for a term T the higher rank this term will assign, and the longer the distance the lower rank this term will receive. So, the first step in ranking is to convert the distance into a score value (T_{DIS_SCORE} or O_I) ranges between zero and one, this incorporate normalizing the result by applying equation 3.6, in this equation the smaller the T_{MIN}, the greater the T_{DIS_SCORE} is.

$$T_{DIS_SCORE} = O_I = 1 - \frac{T_{MIN}}{T_{MAX}}. \tag{3.6}$$

where T_{MIN} is the term minimum distance in the currently under-processing web-page, and T_{MAX} is the largest minimum distance calculated for that term across all pages. For the term v6615en, the minimum distance for the image is 716, and the maximum distance is 46340.0, applying equation 3.6 produces a score of 0.984549, note that this result is the same as O_I in equation 4.1. Using the same manner we take the score of the external search engine using the ordering of images it returns after the search, and

the score is normalized also on the period between 0 and 1.0 using equation 3.7.

$$E_I = 1 - \frac{I_{index}}{count}. \tag{3.7}$$

where I_{index} is the index that the image I comes in the search engine result, and count is the total number of processed result from the search engine, e.g., for our search to term v6615en, search engine returned 108 images, so the first image come with index = 1, and the last image with index = 108, so applying equation 3.7 will produces 1 and 0 for images with index =1, and index =108, respectively.

3.4 Illustration and Discussion

In this section, we will elaborate on steps of the proposed algorithm through an illustration example. First we use Google image search engine to search for a laptop using its model number v6615en. The search is carried out without any constraints on the result set and while the Safe Search option is turned off. Figure 3.11 shows the result

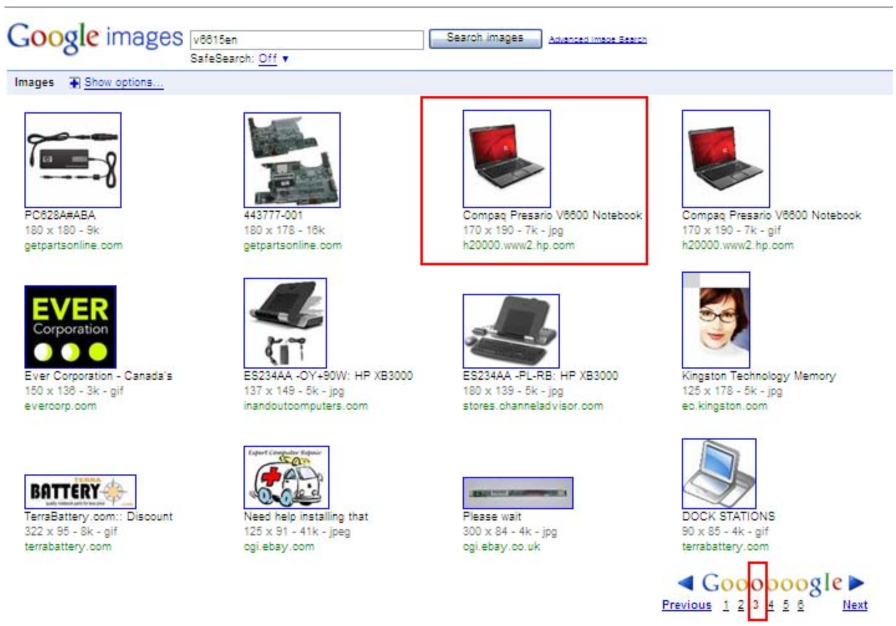


Figure 3.11: *Google result for term v6615en*

returned by Google[2] to search for the laptop. The results show no correct image in the first two pages (each page contains 18 images) of images returned from Google. The correct image of the laptop, which appears on Figure 3.12 as the result number four, is surrounded by a rectangle.

Table 3.3: *Sample of results after Data Retrieval Module for the term v6615en*

ID	TERM	WP_{URL}	RP_{URL}
7069	v6615en	`http://www.terrabattery.com/battery.html`	`http://www.terrabattery.com/235501_1203720208.jpg`
7070	v6615en	`http://www.ramshopping.fr/PC2-5300_HP-Compaq_Presario.html`	`http://www.ramshopping.fr/compaq-presario.gif`
7071	v6615en	`http://h20000.www2.hp.com/ProductList.jsp?taskId=135`	`http://bizsupport2.austin.hp.com/c00668256/c00668256.jpg`
7072	v6615en	`http://forums.eslcafe.com/korea/viewtopic.php?t=158659`	`http://i536.photobucket.com/11494_Apple_Temp_Warning.jpg`
7074	v6615en	`http://stores.channeladvisor.com/item.aspx?Fitemid=800699`	`http://images.channeladvisor.com/tn4_550541.jpg`

First, the Data Retrieval Module harvests the results from Google Image search engine, when querying Google it returns only 108 images. We simulate an Internet browser agent to communicate with Google servers because Google refuses a direct communication. After result received at harvesting step, the harvester saves all results returned by Google after parsing them and extract both the RP_{URL} and WP_{URL}. Next, the renderer takes its turn, starting by traversing results saved from harvester, and rendering the WP_{URL} to produce WP_C. At the same time, the system downloads all RP_{URL} images, Figure 3.4 represent an example of one of the WP_C and RP_{URL}. Table 3.3 represent the output after completing Data Retrieval step.

Second, The Location Determination Module start processing results of the previous module, it extracts the locations of all RP_{URL} inside each page of the 108 pages, as explained in Section 3.2. Also, the location of the search query text v6615en in all 108

[2]This result was taken on October, 16th 2009

pages is determined by executing script discussed in Section 3.2.2. Table 3.4 represent a sample of term location. Table 3.5 also shows a sample of the result of extracting images location for the term v6615en.

Table 3.4: *Sample of terms location*

ID	XPIXEL	YPIXEL	TAG
7070	12982	128	A,
7071	353	915	A,TD,TR,TBODY,
7072	2438	360	SPAN,TD,TR,TBODY,

Table 3.5: *Sample of Image locations*

ID	lo_x	hi_x	lo_y	hi_y
7070	54	142	16958	16870
7071	308	412	1066	950
7073	72	220	1342	1194

Finally, The Re-ranking Module receives the result, and apply the ordering algorithm to come up with the new ordering, the Re-ranking Module is supposed to apply geometric distance measurement as in Section 3.3.1, this will produce the final measurement of distances for all images. Table 3.6 depicts a portion of the database table resulted by our system. This portion shows a descending ordering of the images based on distance or score. Note that in our system, the shorter the distance is the more relevant the image is to the search query. The table contains several columns. $Term$ column is the searched keyword, and the MIN column is the minimum distance between the term and RP_{URL}. The MIN field is populated by applying distance measurement and calculation as explained in section 3.3.1. Based on the Tables 3.4, 3.5 and 3.6 it is clear that the keyword lies on the bottom most right region in Figure 3.10, i.e., the distance is measured using the formula $(c_x - hi_x)^2 + (lo_y - c_y)^2$.

To produce the net distance value (716), this value appears in MIN column in the same row as the processed image with the ID of 7071. DIS_SCORE column in Table 3.6 represents the final re-ranking of each image and is calculated by dividing the minimum

Table 3.6: Sample of result after re-ranking

ID	TERM	WP_{URL}	MIN	DIS_SCORE
7070	v6615en	`http://www.ramshopping.fr/PC2-5300_HP-Compaq_Presario.html`	17029	0.632521
7071	v6615en	`http://h20000.www2.hp.com/ProductList.jsp?taskId=135`	872	0.981183
7072	v6615en	`http://forums.eslcafe.com/viewtopic.php?t=158659`	2464	0.946828

distance for the term T by the maximum distance for that term as in equation 3.6. After re-ranking all 108 images returned by Google, the laptop image that appears in Google on page 3 appears the fifth result based on our technique, see Figure 3.12. Figure 3.12 shows the result of re-ranking images for the term v6615en, this screen also used to tag the number of irrelevant images per block, each row consists of five images, beside each row we have a text box to enter the number of irrelevant images, to be saved and processed later.



Figure 3.12: *Result after re-ranking with our system*

Chapter 4

Results and Discussion

This chapter discusses the results of the conducted experiments to verify the physical distance ranking algorithm. We first present the experiments' methodology. We then present results of experiments.

4.1 Experimental Methodology

We choose two sets of queries to validate our algorithm. The first one relates to searching for products using specific models. The second set concerns searching for animals by their names. The selection of products is due to fact that when a user searches for a product, user looks for one specific thing, which makes the validation of the result more easier. We also test the behavior of the system on a different domain, which is searching for animals.

Experiments start by traversing the top N results from Google image search engine. The setup for searching for products and animals is initially set to allow Google to search without any restriction on the domain of search. In all experiments, the Safe Search is turned off. After we collect the initial data, we apply the algorithm steps previously explained in Chapter 3. After the Re-ranking step, we obtain the new ordering.

We follow the same rules when measuring the improvement and quality of results of our algorithm as in Page-rank for Product Image Search [33] . We display the results returned from Google along with our modified re-ranked results on the same page in

blocks of five images. We ask a human subject to compare between the blocks of pictures of the two sets and to count the number of irrelevant or the least relevant images, like what authors of [33] do. For each term, the human subject compare up to four blocks (20 images) between Google's ranking and our ranking. Table 4.1 provide an example of four product models.

Table 4.1: *Number of irrelevant images for four product models*

Term	No. of irrelevant images in first 5 images	No. of irrelevant images in first 10 images	No. of irrelevant images in first 15 images	No. of irrelevant images in first 20 images
3589i	1	4	6	7
cg180	0	2	4	7
CP-375	4	7	11	16
CX-96	0	1	3	5

Another experiments is also conducted but with a different setup that restricts Google to search within a specific domain. We apply this restriction only to the animals set. We built a user interface to display the re-ranked images along with results coming from Google to make it easy to compare results. Aside from the two experiments cases, we also re-rank the returned images by employing a weighted average of Google ranking and our ranking as follows. Let Google rank for image I be E_I and our ranking for the same image, I, be O_I. Then the new composite ranking C_I is given by

$$C_I = \alpha \cdot O_I + (1 - \alpha) \cdot E_I, \tag{4.1}$$

where α is a tuning factor between zero and one. When α is zero, the ranking is pure Google's we omit our ranking, and when it is one, then we fully incorporate our ranking, but we can not say that we do not incorporate Gogol's rank as the result is initially harvested from Google. A variation between the two extremes (from zero to one), varies the contribution of O_I on the final ranking.

In our experiments we vary α from zero to one in a step of 0.2. For each of the six values of α we display the re-ranked images and ask the user to count the number of

irrelevant images in the first five, ten, fifteen and twenty ordered images, all irrelevant images count from both sets were averaged over the number of queries used, so the result of product set were averaged over 35 queries, and the animal set over 10 queries.

4.2 Product Results

Table 4.2 lists the queries used in our experiments with the associated product set; for each term in the list we provide a small product description. We collected the top 300 ($N = 300$) Google image results for each query to run and validate our algorithm.

Table 4.2: *Product query set*

Product	Description	Product	Description	Product	Description	Product	Description
3589i	Mobile phone	F557US	Laptop	P505-S8945	Laptop	v6615en	Laptop
cg180	Mobile phone	CQ60-410US	Laptop	PDCB-N	Camera Bag	VGN-S48CP	Laptop
CP-375	Television	GA416W-JSA	Remote Control	PMDCB	Camera Bag	VGN-SZ230P	Laptop
GC89	Wireless LAN PC Card	DAH15-00i	MP3 players	PPCS5-00D	Power supply	VX270	PC
CX-96	AM/FM Digital Radio	LX6810-01	PC	PRO2-00X	Data show	DSC-T2L	Camera
mv540	Monitor	MC234-LL	Laptop	SD950	Camera	WS-8349U	Digital Atomic Wall Clock
fx-220	Calculator	DF03A-12	Mini Dome Camera	SV-350	Optical Telescop	XTR-695	Radar detector
w220	Mobile phone	NV52-16u	Laptop	T4200	Laptop		
EZ-46	Remote Control ARF RC Airplane	P2010H	Monitor	TP-750	Power supply		

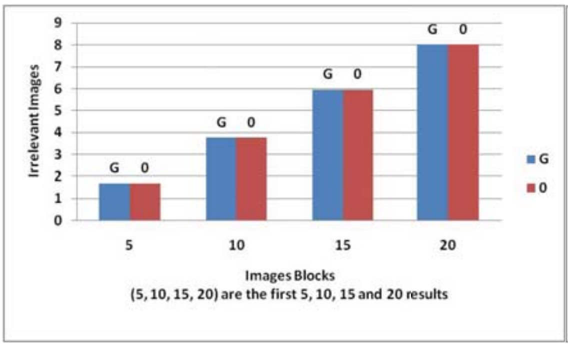
Figure 4.1 shows the results obtained from our system for both α equals zero and 0.2. It is clear from figure 4.1a that for α = zero, we ignore our system's ranking, and so the results will be those of Google's ranking. Figure 4.1b shows a noticeable

improvement to ranking as the number of irrelevant images in first 20 results shrinked, i.e., the improvement is about 15%.

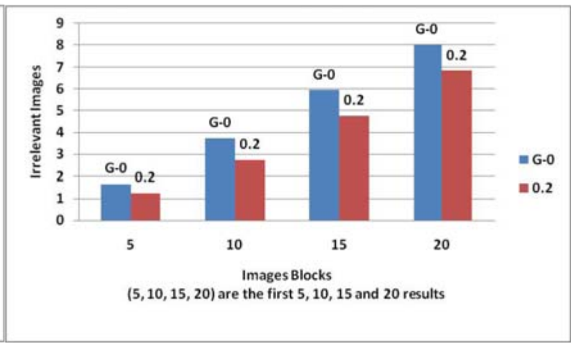
We increase α gradually by applying equation 4.1. The more the value of α is, the more that we incorporate our ranking algorithm and the less that we take Google ranking into account. Figures 4.2 shows a considerable improvement for $\alpha = 0.4$ and 0.6. For example, the number of irrelevant in first 20 results decreases by a percentage of 19% for $\alpha = 0.4$ and 27% for $\alpha = 0.6$. Figure 4.3, respectively complements previous results for values of $\alpha = 0.8$ and 1.0. Figure 4.4 groups all the above Figures to simplify comparisons between them. This Figure shows the following:

- the zero histogram represent Google ranking, as equation 4.1 states.
- for all four blocks, our algorithm always reduces irrelevant images and improves ranking.
- on average, $\alpha = 0.8$ proves the best performance with a small difference than $\alpha = 1$.

Figure 4.5 shows the percentage of error reduction ($Error_{percentage}$). We compute this percentage by dividing the difference of irrelevant images number between the original Google results($G_{irrelevant}$) and the re-ranked results ($O_{irrelevant}$), and then divide the

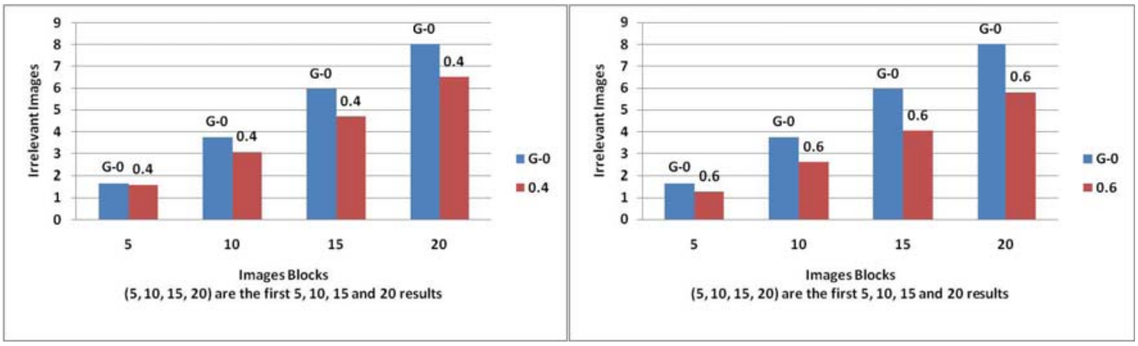


(a) Product irrelevant images for α=0



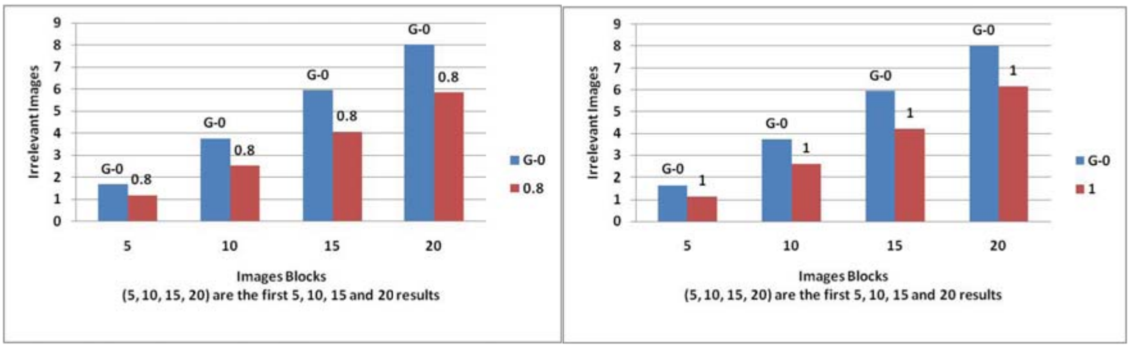
(b) Product irrelevant images for α=0.2

Figure 4.1: *Product irrelevant images for α=0 and 0.2.*



(a) *Product irrelevant images for α=0.4* ***(b)*** *Product irrelevant images for α=0.6*

Figure 4.2: *Product Irrelevant images for α=0.4 and 0.6.*



(a) *Product irrelevant images for α=0.8* ***(b)*** *Product irrelevant images for α=1.0*

Figure 4.3: *Product irrelevant images for α=0.8 and 1.0.*

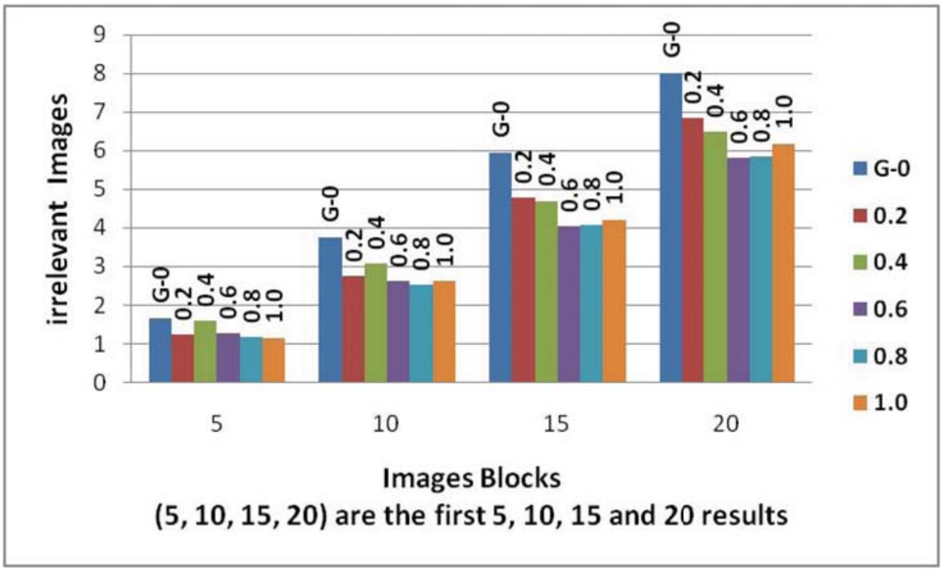


Figure 4.4: *Product irrelevant images for all variation of α.*

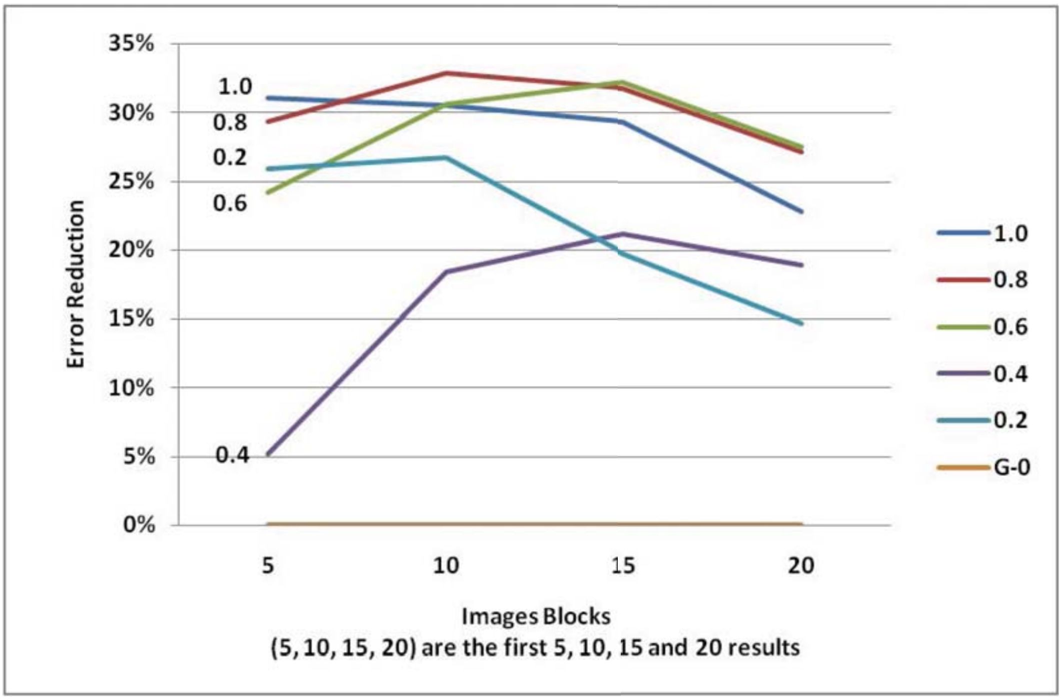


Figure 4.5: *Error reduction percentage for product per each block.*

difference on the Google irrelevant images number, see equation 4.2.

$$Error_{percentage} = \frac{G_{irrelevant} - O_{irrelevant}}{G_{irrelevant}} \tag{4.2}$$

From Figure 4.5, we infer the following:

- the zero percentage happens only when α equals zero.
- the maximum improvement happens when $\alpha = 0.8$, in first 10 results.
- if we care about improvement to first block which represents the top re-ranked images, it score the best percentage for α equals one.
- if we care about overall cumulative performance, then we choose $\alpha = 0.6$ and 0.8.

4.3 Animals Results

Table 4.3 lists the queries used in our experiments with the animal set. We collect the top 100 results from Google's image search engine for each term in Table 4.3 (N=100), and we then apply our algorithm steps. In this case, the results returned by Google is from the whole Web (i.e., open domain). After ranking is applied and displayed to the user, the user tags the irrelevant images for different values of α as before.

Table 4.3: *Animals query set*

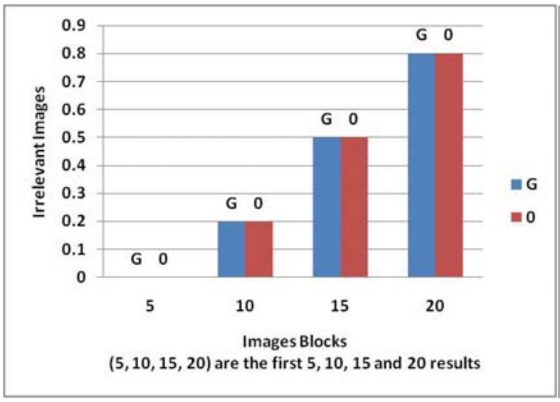
Anglerfish	A fish famous for its extreme ugliness
ant	Social insect living in organized colonies
anteater	Toothless mammal with body covered with horny scales and a long snout for feeding on ants and termites
monkey	Any of various long-tailed, medium-sized members of the order Primates
baboon	Large terrestrial monkeys
bobcat	Small lynx of North America
caribou	Arctic deer with large antlers in both sexes
Coyote	Small wolf native to western North America
snake	Elongate legless carnivorous reptiles of the suborder Serpentes
Ammonites	Well known and most common fossil found and collected all over the world

4.3.1 Open domain

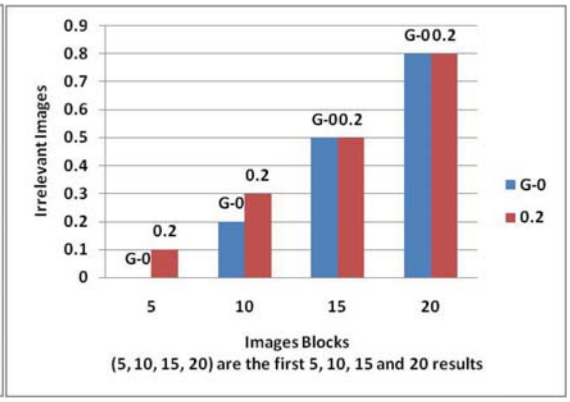
Figure 4.6 shows the results for α = zero and 0.2. The results are almost identical for the two cases.

Figures 4.7a, 4.7b, 4.8a and 4.8b shows the improvment of our ranking algorithm over Google ranking.

Figure 4.9 shows an improvement between 13% and 25% in the first 15 and 20 results, respectively.

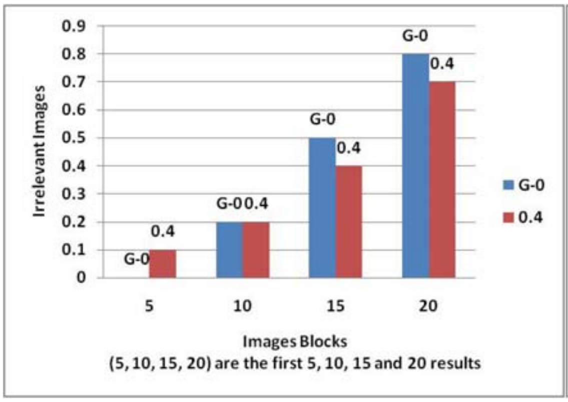


(a) *Animals irrelevant images for* α*=0*

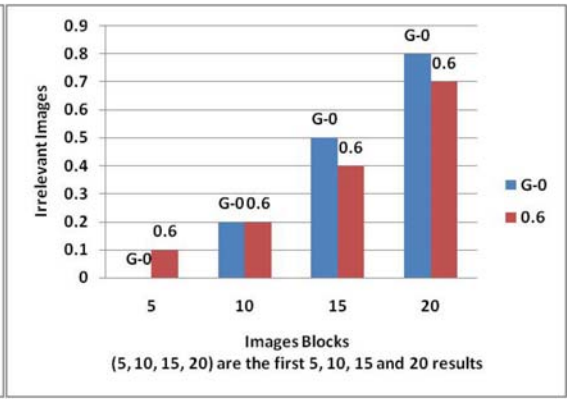


(b) *Animals irrelevant images for* α*=0.2*

Figure 4.6: *Animals irrelevant images for* α*=0 and 0.2.*

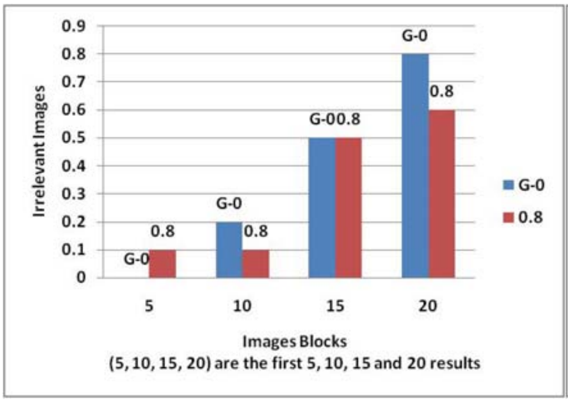


(a) *Animals irrelevant images for* α*=0.4*

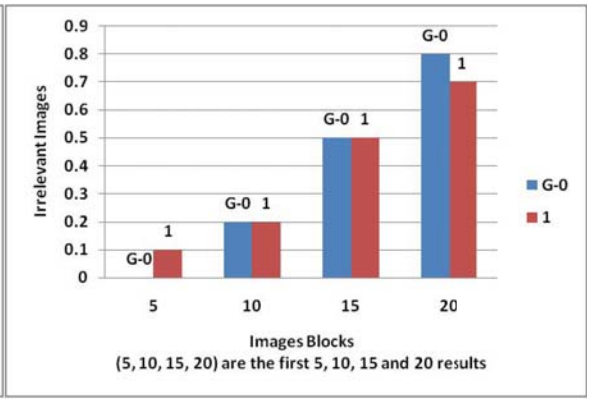


(b) *Animals irrelevant images for* α*=0.6*

Figure 4.7: *Animals irrelevant images for* α*=0.4 and 0.6.*



(a) *Animals irrelevant images for* α*=0.8*



(b) *Animals irrelevant images for* α*=1.0*

Figure 4.8: *Animals irrelevant images for* α*=0.8 and 1.0.*

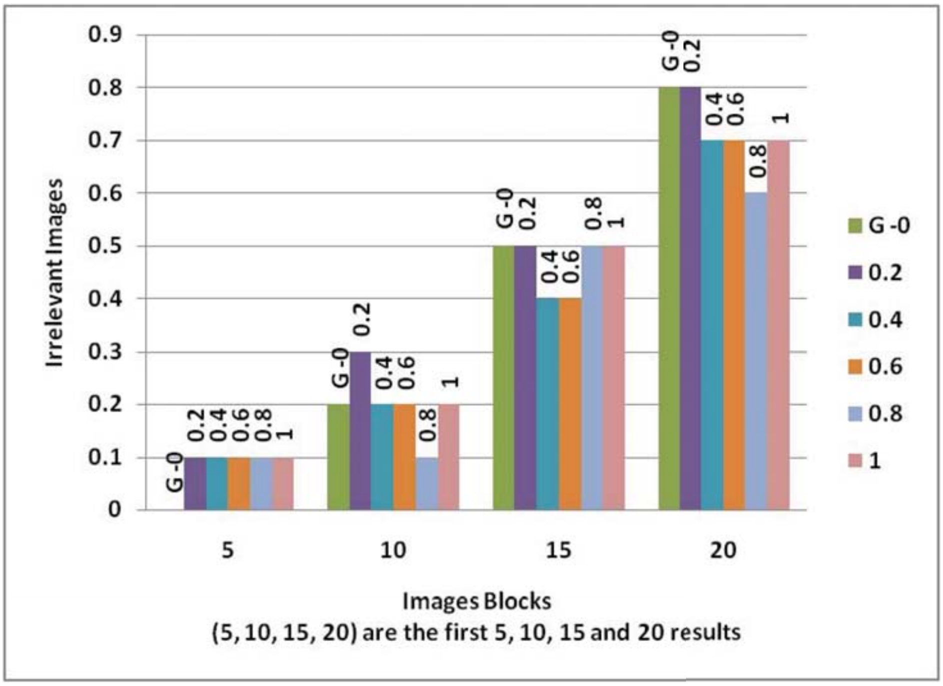


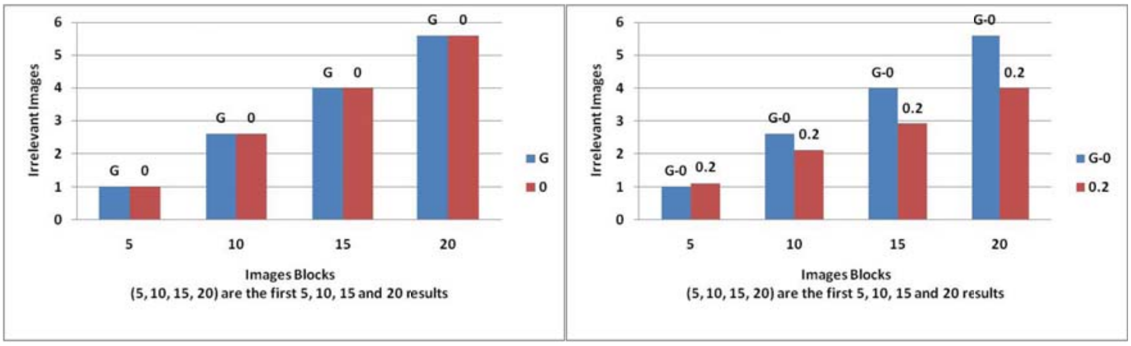
Figure 4.9: *Animal irrelevant images for all variation of α.*

4.3.2 Closed domain

In this section, we harvested the top 100 images from Google image search engine but now we restrict Google to search in specific domain, which is `wikimedia.org`. We use the same query set in Table 4.3. The results show a better improvement of our algorithm over Google than that of section 4.3.1. Figure 4.10b shows a significant improvement after applying our ranking algorithm with the improvement in first 5, 10, 15 and 20 results varies from 0% to 30%.

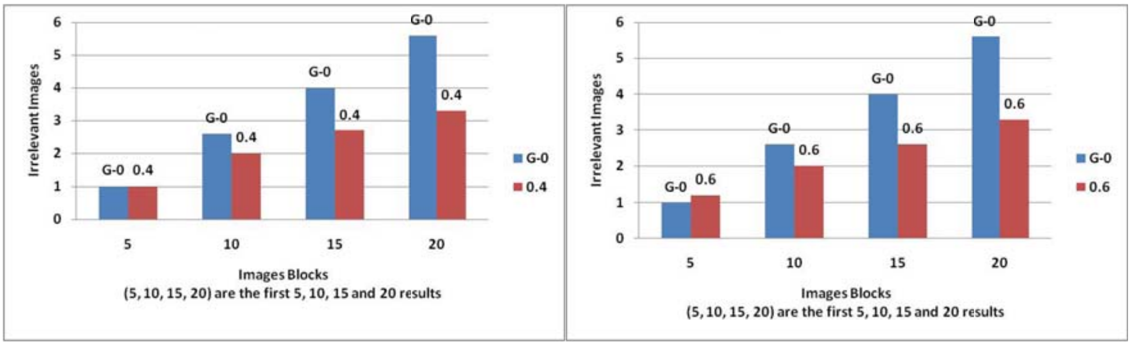
Figure 4.11a and 4.11b shows the maximum improvement on this set with $\alpha = 0.4$ and 0.6. Our ranking algorithm achieves a total of 41% improvement in the first 20 results. This is the case also in Figure 4.12, where it shows an improvement of 40% and 33% for $\alpha = 0.8$ and 1 respectively.

Figure 4.13 shows a combined view of all previous figures, it shows that with $\alpha = 0.4$, the system achieves the maximum improvement, on the other hand, the system with various α values achieves improvement in the range of 28% to 39%.



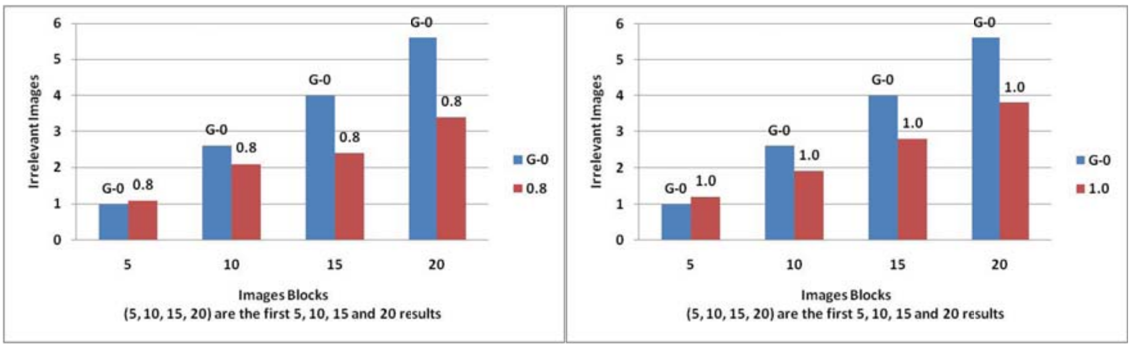
(a) Wikimedia irrelevant images for α=0 *(b) Wikimedia irrelevant images for α=0.2*

Figure 4.10: *Wikimedia irrelevant images for α=0.2 and 0.4.*



(a) Wikimedia irrelevant images for α=0.4 *(b) Wikimedia irrelevant images for α=0.6*

Figure 4.11: *Wikimedia irrelevant images for α=0.4 and 0.6.*



(a) Wikimedia irrelevant images for α=0.8 *(b) Wikimedia irrelevant images for α=1.0*

Figure 4.12: *Wikimedia irrelevant images for α=0.8 and 1.0.*

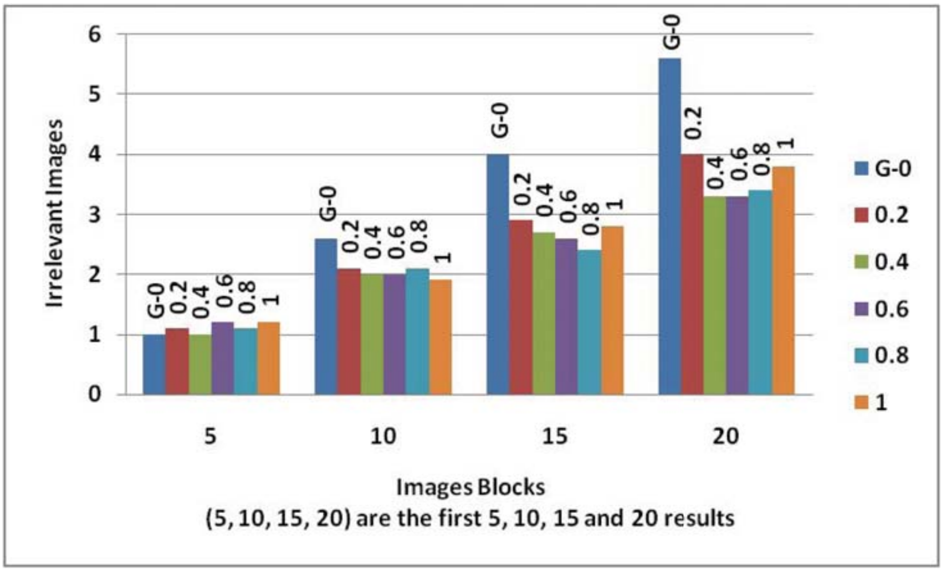


Figure 4.13: *Wikimedia irrelevant images for all variation of* α.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Our work aims to enhance the ordering and ranking of image search method by taking into consideration the physical distance between an image and a keyword inside a web page. Our algorithm relies on strong semantic relation between an image and the surrounding text. Our algorithm achieves a considerable improvement over the well-known, highly reputed image search engine, Google.

We choose two data sets of queries to validate our algorithm. The first one targets products model number, and the second is a set of animal names. Choosing product models as queries makes the process of categorizing images relevancy more easier since a product model is a unique identifier of a product and the chance of wrong categorization is minimized. The a second set is used to test the behavior of the system on other well known domains so we used queries that target the animals domain.

The steps encapsulated into our work to incorporate distance into ranking images on the web are divided into three main steps. The Data Retrieval Module is the first component to be integrated to perform the initial data gathering. It queries the search engine and finds the suggested resultant images. The second step is the Location Determination Module, which extracts the location of the resultant image from the previous step, and extracts the text locations.

Finally, the Ranking module applies a simple ranking algorithm based on the calculated distances and scores. The ranking system then displays the final results to the

user who is responsible to classify irrelevant images

The ranking algorithm can be tuned to change the contribution of our physical-distance ranking on the final displayed results using the parameter called α and according to equation 4.1. Varying the parameter varies the final ranking produced from the system

5.2 Future Work

The execution performance of our system was not of our major concern here. We only concentrated to proof the concept of relating physical distance to surrounding near text inside a web-page, and we never cared about the processing time required and the amount of allocated resources .

We believe that a considerable improvement of processing time can be achieved, especially if we target the critical part of the system that consumes most of processing time and drains resource, which is rendering step that determines an image's location inside a web page. This will ensure a simple, light and utilized integration of our algorithm into any existent image search engine.

Bibliography

[1] Mac OS Reference Library, Search Kit Programming Guide; 2009. Available from: http://developer.apple.com/mac/library/documentation/UserExperience/Conceptual/SearchKitConcepts/searchKit_basics/searchKit_basics.html.

[2] Cai D, Yu S, Wen JR, Ma WY. Extracting Content Structure for Web Pages Based on Visual Representation; 2003. 596. Available from: http://www.springerlink.com/content/a65708xmur9kn9ea.

[3] Shieh JR, Yeh YT, Lin CH, Lin CY, Wu JL. Using Semantic Graphs for Image Search. In: Multimedia and Expo, 2008 IEEE International Conference on; 2008. 105–108. Available from: http://dx.doi.org/10.1109/ICME.2008.4607382.

[4] Veltkamp RC, Tanase M. Content-based image retrieval systems: A survey; 2000. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.7344.

[5] Vlachos M, Vardangalos G, Tatsiopoulos C. Effective ways for querying images by content over the Internet; 2000.

[6] Introduction to Data Structures; 2009. Available from: http://www.cs.umd.edu/~mount/420/Lects/420lects.pdf.

[7] Leading worldwide market research and strategic consulting firm for the digital imaging and document solutions industry; 2009. Available from: http://www.infotrends-rgi.com.

[8] Advanced data collection methodologies and measurement science company; 2009. Available from: http://www.nielsen.com.

[9] Helms D, Schmidt E, Nebel W. Leakage in CMOS Circuits; 2004. 17–35.

[10] Datta R, Joshi D, Li J, James, Wang Z. Image retrieval: Ideas, influences, and trends of the new age. ACM Computing Surveys. 2006;39:2007.

[11] Cutler M, Deng H, Maniccam SS, Meng W. A new study on using HTML structures to improve retrieval. In: Tools with Artificial Intelligence; 1999. 406–409. Available from: http://dx.doi.org/10.1109/TAI.1999.809831.

[12] Hatcher E, Gospodnetic O. Lucene in Action. Manning Publications; 2004. Available from: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1932394281.

[13] Rui X, Yu N, Wang T, Li M. A Search-Based Web Image Annotation Method. In: Multimedia and Expo, 2007 IEEE International Conference on; 2007. 655–658. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4284735.

[14] Rinc S. Towards Standard Semantic Image Annotation And Search. Content-Based Multimedia Indexing. 2008; 484–488.

[15] Gao S, Chevallet JP, Lim JH. Rich Representation and Ranking for Photographic Image Retrieval in ImageCLEF 2007; 2008.

[16] Cail D, Yu S, Wen JR, Ma WY. Block-based Web Search. ACM; 2004. .

[17] Salton G, Allan J, Buckley C. Approaches to passage retrieval in full text information systems. In: SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. New York, NY, USA: ACM; 1993. 49–58. Available from: http://dx.doi.org/10.1145/160688.160693.

[18] Powell TA. HTML and XHTML: The Complete Reference (4th Edition). McGraw-Hill; 2003.

[19] Tatiana, Calado PP, Souza LV, Neto BR, Muntz R. Image Retrieval Using Multiple Evidence Ranking. IEEE Transactions on Knowledge and Data Engineering. 2004;16(4):408–417. Available from: http://dx.doi.org/10.1109/TKDE.2004.1269666.

[20] Wnuk k, Soatto S. Filtering Internet Image Search Results Towards Keyword Based Category Recognition. IEEE; 2008. 1–8.

[21] Lu G, Williams B. An Integrated WWW Image Retrieval Syste. Fifth Australian World Wide Web Conference; 1999.

[22] Yao J, Li M, Li Z, Zhang L, Ma WY. Natural Image Retrieval With Sketches. In: SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. ICME; 2005. 1198–1201.

[23] Barthel KU. Improved Image Retrieval Using Automatic Image Sorting and Semi-automatic Generation of Image Semantics. Image Analysis for Multimedia Interactive Services. 2008; 227–230.

[24] Bendersky M, Kurland O. Re-Ranking Search Results Using Document-Passage Graphs. In: SIGIR '08;. .

[25] Russell S, Norvig P. Artificial Intelligence: A Modern Approach. 2nd ed. Prentice Hall; 2002. Available from: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0137903952.

[26] Mukherjea S, Hirata K, Hara Y. AMORE: A World Wide Web Image Retrieval Engine. World Wide Web. 1999;2(3):115–132. Available from: http://dblp.uni-trier.de/db/journals/www/www2.html#MukherjeaHH99.

[27] Aurnhammer M, Hanappe P, Steels L. Integrating Collaborative Tagging and Emergent Semantics for Image Retrieval. In: Proceedings WWW2006, Collaborative Web Tagging Workshop; 2006. Available from: http://www.isrl.uiuc.edu/~amag/langev/paper/aurnhammer06semanticsWWW.html.

[28] Hugh W. Ranking Images for Web Image Retrieval; 2008.

[29] Viewing HTML pages with SWT Browser widget; 2009. Available from: http://www.eclipse.org/articles/Article-SWT-browser-widget/browser.html.

[30] Gonzalez RC, Woods RE. Digital Image Processing (3rd Edition). Prentice Hall; 2007. Available from: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/013168728X.

[31] Kumar V. Correlation Pattern Recognition. Cambridge University Press; 2005.

[32] Marini J. The Document Object Model: Processing Structured Documents. McGraw-Hill; 2002.

[33] Jing Y, Baluja S. PageRank for Product Image Search. In: In: WWW 2008. Refereed Track: Rich Media; 2008. .

[34] Kushchu I. Web-based evolutionary and adaptive information retrieval. Evolutionary Computation, IEEE Transactions on. 2005;9(2):117–125. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1413254.

[35] Inverted index; 2009. Available from: http://en.wikipedia.org/wiki/Inverted_index.

تحسين البحث عن الصور في شبكة الانترنت بأدوات نصيه

اعداد: رامي عبدالوهاب الهيلات

الملخص

كشفت الدراسات الحديثة عن وجود أكثر من 300 مليون جهاز لالتقاط الصور الرقمية في جميع أنحاء العالم [1] يتم استخدامها لالتقاط حوالي 29 مليارمن الصور الرقمية . معظم هذه الصور تنظم وترتب في مستودعات رقميه خاصه بالوسائط المتعددة ، ويتم نشرها ومشاركتها باستخدام الشبكة العنكبوتية.
كثير من هذه الصور تحتاج إلى التخزين والفرز وغالبا الاسترجاع فيما بعد. لذا فأن الحاجة ضروريه لوجود منهج واضح لمساعدة المستخدمين الاعتياديين لهذه الصور للبحث فيها واسترجاع المطلوب منها بسهوله باستخدام الانترنت، خصوصا اذا علمنا أن ما بين 15 ٪ و 16 ٪ من عمليات البحث على الإنترنت هي للصور [2].

تم اختراع العديد من الطرق للبحث والاسترجاع على الانترنت, معظم هذه الطرق كانت تستخدم السياق النصي لصفحات الانترنت للتمييز بين الصور التي لها علاقه بكلمة البحث والصور الأخرى التي ليس لها علاقه. على صعيد اخر هناك طرق أخرى كانت تستغل المحتويات والخصائص الداخليه للصور في عملية البحث و من الأمثلة على هذه الخصائص ,الرسم البياني, البنيه النسيجيه للصوره و أنماط واصفات الحدود. ان استخدام أي من هذه الخصائص له بعض المزايا و العيوب أيضا.
أن مقترح هذه الرسالة يهدف لتحسين البحث عن الصور من خلال تطبيق أسلوب جديد يستخدم المسافة الفاصله بين صورة في صفحة الويب و النصوص المحيطة بها كمقياس لمدى العلاقه بينهما . و قد أظهر هذا المقترح تحسينا كبيراعلى عملية البحث وترتيب نتائجها.